OPERATOR'S MANUAL

BE-64

Bus Emulator Demo Board

Manual Revision: 01/10/08
Manual Part Number: BEOM112
Instrument Part Number: BEOM112

talon
INSTRUMENTS
An EADS North America Defense Company

CERTIFICATION

Talon Instruments certifies that this product met its published specifications at the time of shipment from the factory.

WARRANTY

Talon Instruments products are warranted against defects in materials and workmanship as follows:

(a) One year for the MA1801.

(b) Ninety days for cables.

During the warranty period, Talon Instruments will, at its option, either repair or replace products which prove to be defective.

For warranty service or repair, this product must be returned to the Talon Instruments factory. Buyer shall prepay shipping charges to the factory and Talon Instruments shall pay shipping charges to return the product to the Buyer. However, Buyer shall pay all shipping charges, duties, and taxes for products returned to Talon Instruments from another country.

Talon Instruments warrants that its software and firmware designated by Talon for use with its instruments will execute its programming instructions when properly installed on the instrument. Talon Instruments does not warrant that the operation of the instrument, or software, or firmware will be uninterrupted or error free.

LIMITATION OF WARRANTY

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by the Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of environmental specifications for the product, or improper site preparation or maintenance.

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED. TALON INSTRUMENTS SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

EXCLUSIVE REMEDIES

THE REMEDIES PROVIDED HEREIN ARE BUYER'S SOLE AND EXCLUSIVE REMEDIES. TALON INSTRUMENTS SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER BASED ON CONTRACT, TORT, OR ANY OTHER LEGAL THEORY.

SAFETY FIRST

PROTECT YOURSELF AND THE EQUIPMENT.

Follow these precautions:

- Don't repair the module unless you are a qualified electronics technician and have instructions from Talon Instruments.
- Pay attention to the WARNING statements. They point out situations that can cause injury or death.
- Pay attention to the CAUTION statements. They point out situations that can cause equipment damage.
- Use ESD static control procedures when handling the MA1801 or any of its modules.

TABLE OF CONTENTS

LIST OF FIGURES

1 INTRODUCTION

Many of todays digital cards incorporate a “bus structured” architecture. Although the buses, or interfaces, to these boards may differ in various ways, the communication concepts for these boards are quite similar.

The Bus Emulator Demo Board (Fig 1) has been developed to incorporate many of the features found in digital boards with parallel bus interfaces. The intent of this demo board is as follows:

1) Supply an educational tool to teach the art of bus emulation testing.
2) Demonstrate how the Talon’s BE-64 and SR192 VXI boards test a typical bus structured board. This includes programming the hardware (timing sets and field control) as well as programming message based commands to test a typical board.
3) An experimental area is provided such that the user can add other UUT devices (counters, pals, etc) and compare bus emulator test methods with traditional stimulus/response test methods.

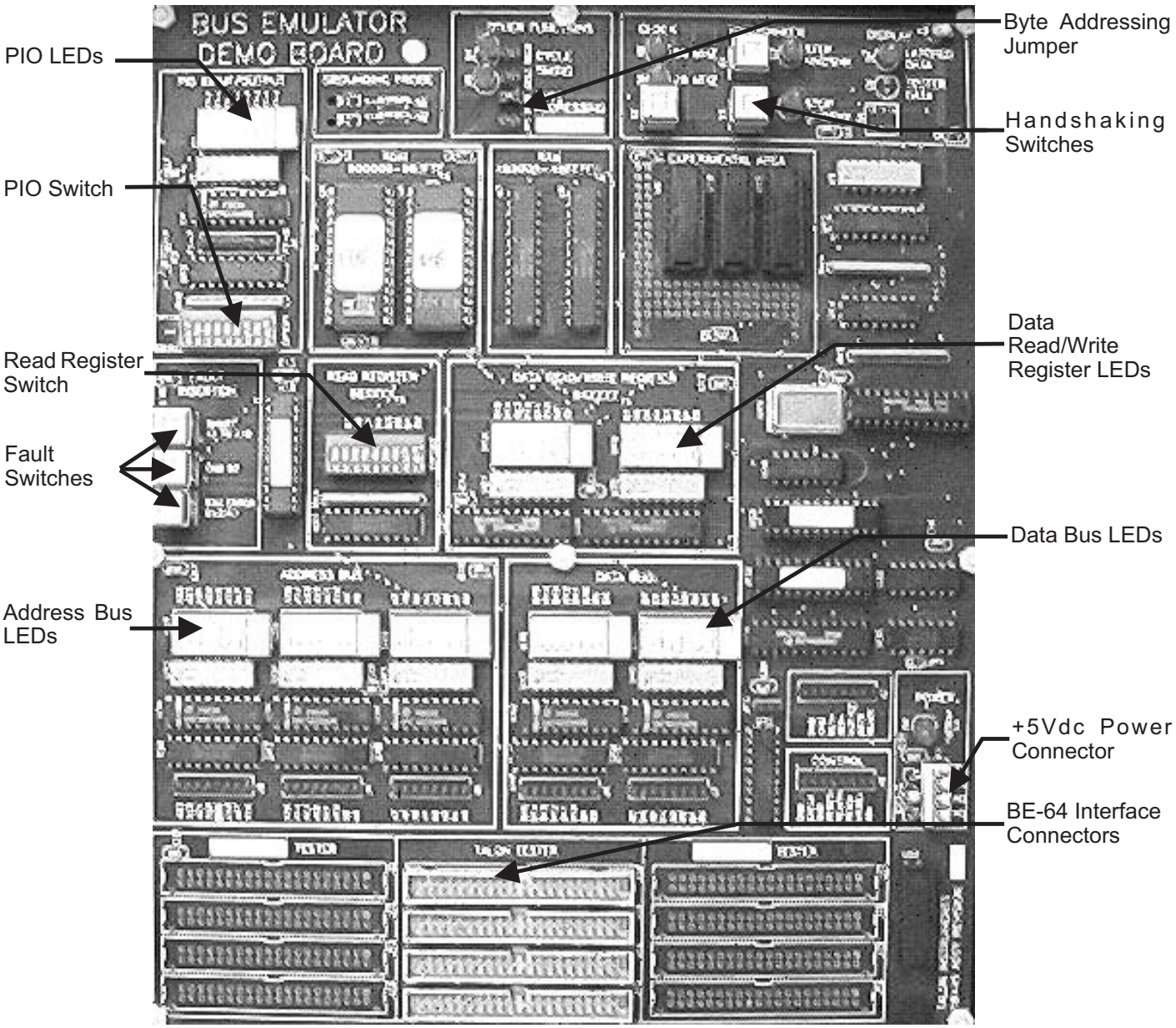


Figure 1-1 Demo Board

2 DEMO BOARD KIT AND SUPPORT REQUIREMENTS

The Demo Board is shipped in a kit containing the items listed below. If any of these items are missing please call Talon or your local representative before proceeding.

2.1 DEMO KIT CONTENTS

1) Demo board along with two interconnect cables for the bus emulator and a set of power cables with banana jacks.
2) Talon VXI Bus Emulator Module.
3) BE-64 Plug&play Drivers
4) BE-64 Timing & Table Editor Software

2.2 USER SUPPLIED ITEMS

1) The hardware required to operate the demo board is a VXI system with controller along with a UUT power supply. The power supply must provide the demo board with +5 volts at 2.5 amps. This power is connected via the P1 connector by use of the power cable shipped with the demo kit. LED D9 will light when +5 volts is properly applied to the board.
2) The VXI system must provide a means of communicating to the bus emulator via word serial command strings. The ability to send/receive word serial strings and files will permit the user to operate the demo board. The VXI controller software must use VXI Plug&play VISA Revision 1.0 or higher.

3 DEMO BOARD DESCRIPTION

3.1 BLOCK DIAGRAM

Figure 3-1 depicts the block diagram for the demo board. The Bus Emulator Demo Board incorporates a ROM, a RAM, a Read Register (8 switches), and a Read/Write Register. A controlling device (i.e. tester) communicates with this logic via the bus interface, including a 24 bit address bus, 16 bit data bus and a control interface. The address and data buses can be visually monitored via 24 and 16 leds respectively.

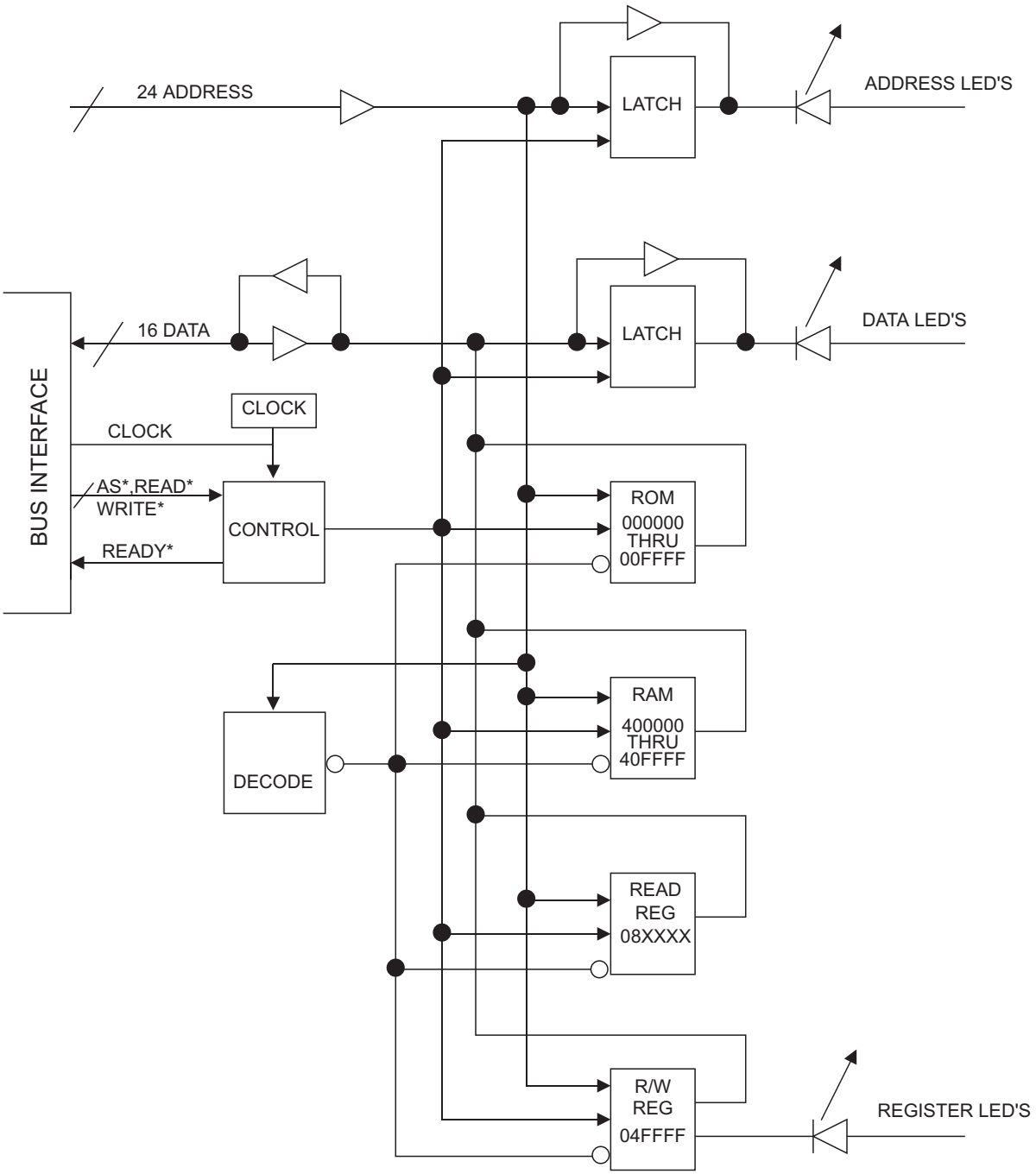
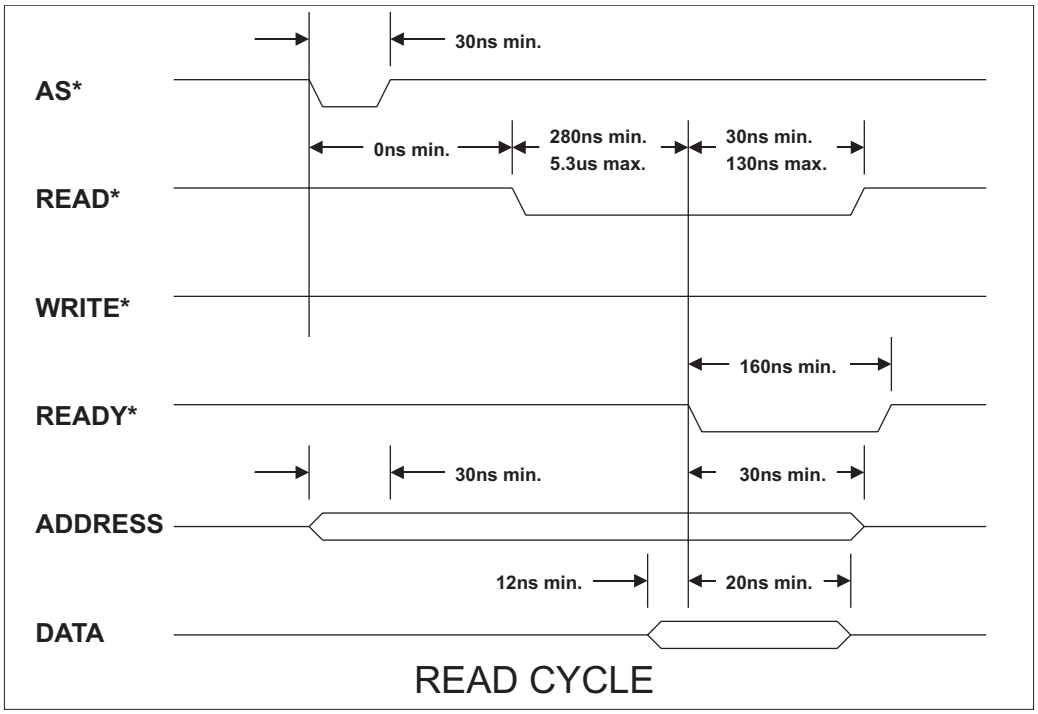
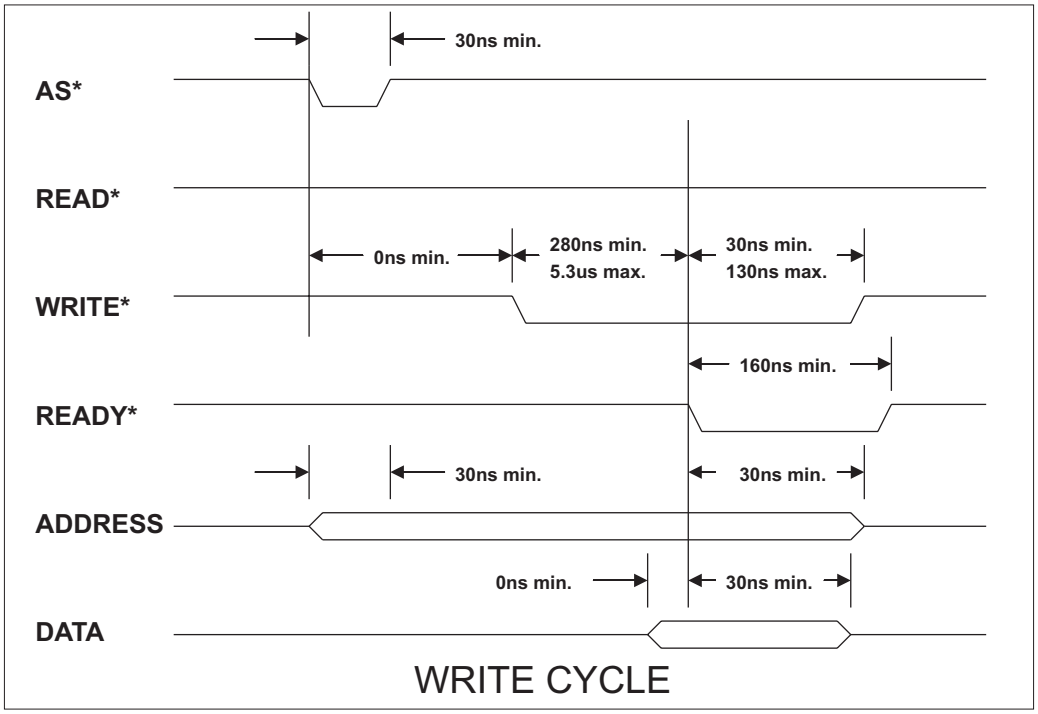


Figure 3-1 Demo Board Block Diagram

3.2 BUS CYCLE TIMING

The following figure illustrate the read and write bus cycle timing for the BE64 demonstartion board.





Timing assumes 50MHz clock. Multiply all times by 2.5 for 20MHz clock selection.

Figure 3-2 Bus Cycle Timing Diagram

3.3 BUS INTERFACE DESCRIPTION

Communication to and from the demo board is accomplished over the address and data buses with each bus cycle controlled by the control bus.

3.4 CLOCK

The Bus Emulator Demo Board is clocked from an internal 50 Mhz oscillator. This clock will generate the bus timing defined by Figure 3.2. For testers which can't handle the 50 Mhz timing, switch S2 will change the clock to 20 Mhz.

The 50 Mhz (or 20 Mhz) clock is tied to the control signal EXCLK. This clock may be used to drive the tester.

3.4.1 CONTROL BUS

The control for each bus cycle consists of four signals. Each bus cycle requires a handshake between the controlling device (the tester) and Bus Emulation Demo Board. The four signals are defined below. The timing for valid read and write bus cycles is defined in Figure 3-2.

AS* (Address Strobe): the address strobe, when asserted low by the controlling device, indicates the beginning of a bus cycle. Actually, the only function of the address strobe is to latch the address into the address latch display register. The Bus Emulator Demo Board will execute a successful bus cycle without the address strobe, however, the latched address cannot be displayed.

READ*(Read Cycle): the READ* signal, when asserted low by the controlling device, indicates that the present cycle is a read cycle (i.e. data from the Bus Emulator Demo Board is gated onto the data bus and transferred to the controlling device).

WRITE* (Write Cycle): the Write* signal, when asserted low by the controlling device, indicates that the present cycle is a write cycle (i.e. data from the controlling device is gated onto the data bus and transferred to the Bus Emulator Demo Board).

READY* (Ready): for READ cycles, the READY* signal, when asserted low by the Bus Emulator Demo Board, indicates to the controlling device that valid data has been placed on the data bus and is ready for reading by the controlling device. For WRITE cycles, the READY* signal, when asserted low by the Bus Emulator Demo Board, indicates to the controlling device that the Bus Emulator Demo Board accepted the data on the data bus.

3.4.2 ADDRESS BUS

The address bus consists of twenty four unidirectional lines. Address values are transmitted from the controlling device to the Bus Emulator Demo Board. Address values must be stable as defined by Figure 3-2.

The Bus Emulator Demo Board is a 16 bit device. Although the Bus Emulator Demo Board may be addressed in 8 bit bytes (Reference section 3.9.1), the board is normally configured for 16 bit word operations (i.e. the jumper called BYTE (BYTE ADDRESSING) in the OTHER FUNCTIONS section is NOT installed)

For 16 bit word transfers, address bit 0 (A0) is not used and the board treats A0 as a "don't care".

3.4.3 DATA BUS

The Data Bus consists of 16 bi-directional data lines. For 8 bit byte transfers (Reference Section 3.9.1) D0 through D7 is the least significant byte and D8 through D15 is the most significant byte.

3.4.4 ROM (Address 000000 thru 007FFF hex)

The ROM consists of a 16K x 16 bit ROM. For 16 bit word transfers, A0 is a "don't care".

Example:

ADDRESS	000000 hex 1st 16 bit word
ADDRESS	000002 hex 2nd 16 bit word
ADDRESS	000004 hex 3rd 16 bit word
ADDRESS	000006 hex 4th 16 bit word

3.4.5 RAM (Address 400000 thru 40FFFF hex)

The RAM consists of a 32K x 16 bit RAM. For 16 bit word transfers, A0 is a “don’t care”.

3.4.6 READ REGISTER (Address 08XXXX hex (X=don’t care)

The Read Register consists of an 8 bit switch. When read, the contents of switch DIPSW2 are placed on data bits D0 through D7.

3.4.7 READ/WRITE REGISTER (Address 04XXXX hex)

The Read/Write Register consists of a 16 bit register. A write cycle at address 04XXXX writes data into the register, a read cycle at address 04XXXX reads data from the register.

3.5 FAULT INSERTION

Several faults may be inserted into the Bus Emulator Demo Board. These faults, inserted with switches S5, S6 and S7, simulate typical faults found on digital boards.

3.5.1 DATA BIT GROUNDED (S6)

Depression of switch S6 grounds data bit D7 of the data bus internal to the Bus Emulator Demo Board.

3.5.2 ADDRESS BUS SHORT (S5)

Depression of switch S5 shorts address bit A7 to address bit A10 on connectors J5 and J4 respectively. This fault directly shorts channels on the respective tester.

3.5.3 RAM ERROR (S7)

Depression of switch S7 simulates an error on data bit 13 internal to the RAM U8. In particular, at each address where A7 is True (high) the data in bit 13 is inverted in the RAM U8.

3.5.4 GROUNDING PROBE (V1, V2)

A grounding probe is provided such that the user can force a ground fault at any pin. The probe pins V1 or V2, incorporates a 5 OHM series resistor to ground which prevents an accidental direct short to +5 volts.

3.6 SINGLE STEP

Each bus cycle requires a “handshake” between the controlling device and the Bus Emulator Demo Board. By pressing switch S3, the STOP HANDSHAKE led will illuminate and the Bus Emulator Demo Board READY * signal will be disabled. Each depression of S1 will then generate a single pulse on the READY* signal in accordance with the timing diagrams, Fig 3 and Fig 4.

3.7 DISPLAY DATA

The address and data LEDs can be set to display either DATA (address and data values taken directly from the Demo Board Data Bus) or LATCHED DATA (address and data values latched by the proper execution of a bus cycle). Switch S4 toggles the LEDs between latched and data bus. It is recommended that the switch be set to the latched data position.

3.8 PIO INPUT/OUTPUT

The function of this logic is to demonstrate the programmed I/O feature of Talon’s BE-64. The BE-64’s PI01 signals are tied to J1, pin 1 thru 8. Switch position 9 for DIPSW1 defines the logic to be set in the input or output mode as referenced to the Demo Board.

Switch 9=OFF=Input into Demo Board (i.e. the BE-64 P101 output data will be displayed on LEDs U1.)

Switch 9=ON=Output from Demo Board (i.e. DIPSW1, positions 1 thru 8, can be read by the BE-64 PI01.)

The user should leave switch 9 in the input mode (Switch 9=OFF) until the BE-64 has been properly configured. However, if this is not done, neither the Bus Emulator Demo Board nor the BE-64 will be damaged.

3.9 OTHER FUNCTIONS

3.9.1 BYTE ADDRESSING

The Byte Addressing allows the demonstration of Talon's unique Byte Enable function when the jumper nomenclated BYTE ADDRESSING (BYTE) is installed, the Bus Emulator Demo Board will operate in byte mode, enabling 8 bit byte transfers. This function requires the connection of the BE0* and BE1 signals on J2.

The following table illustrates the byte enable codes for word and byte transfers

ADDRESS(A0)	WORD Transfer		BYTE Transfer	
	D15 . . . D8 BE1	D7 . . . D0 BE0	D15 . . . D8 BE1	D7 . . . D0 BE0
0	0	0	1	0
1	NA	NA	0	1

3.9.2 CYCLE TIMING

Talon's BE-64 and SR192 have the ability to "handshake" with a unit under test with nanosecond accuracy. You may find that other testers do not incorporate this feature. For testers which cannot handshake, or cannot handshake at speed, the CYCLE TIMING function enables the Bus Emulator Demo Board to modify the cycle timing to match the respective testers capability. Insertion of jumpers DEL0 or DEL1 extends the data hold for read cycles by 1 clock or 2 clocks respectively. Insertion of both jumpers DEL0 and DEL1 extends the data hold time by 3 clocks.

4 Programming the Bus Emulator Setup Memories

In order to stimulate and execute tests on the Demo Board the timing set and data table memories must be programmed. The SCPI commands to perform this setup may be prepared using Talon's Windows-based BE-64 Timing/Table Editor or written to the Bus Emulator card as word serial commands.

If you intend to use the Talon editor routines to create the SCPI setup commands follow the directions in Section 4.1. If you are using word serial commands then proceed at this point to section 7.0 Word Serial Command Setup.

4.1 TALON FILE EDITOR SOFTWARE

The Talon BE-64 Timing/Table Editor should be loaded in your PC following the instructions contained in the operations manual sent with the software disk. Start the Timing Cycle setup routines by clicking on the timing editor icon.

BE-64 Timing Editor

4.1.1 PROGRAMMING TIMING SET MEMORY

The demo board requires two timing cycles to be defined. Since we wish to use the Bus Emulator's automatic RAM and ROM tests, the two cycles to be programmed are the READ_MEM and WRITE_MEM cycles. Figures containing the timing data for the read and write cycles are shown in section 3.2 of the previous chapter.

The global setup commands should be programmed first. Click on the Timing Sets menu and select the SETUP command. The setup area allows the user to change the default BE-64 signal names to the UUT signal names to aid in programming the timng sets. The Clock Source should be set to External, Time-out to 1000 and Test Delay to 32,767.

The following signal names may be entered to replace the default mnemonics for ease of use:

Default Mnemonic	UUT Signal Name
TSOUT1	AS*
TSOUT2	WRITE*
TSOUT3	READ*
TSOUT8	BYTE_EN
Enable Field 1	EN_ADDR
Enable Field 2	EN_DATA
TSSTROBE	READY*

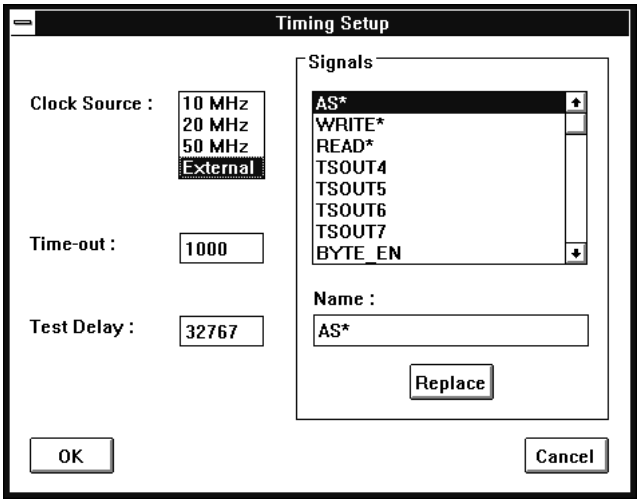


Figure 4-1 Timing Sets SETUP Window

Individual timing cycles may now be programmed. Use the Timing Sets menu to select the View command. The

READ_MEM cycle will be programmed first. The individual cycle parameters are programmed by using the Timing Sets menu to select the Parameters window. See Figures below.

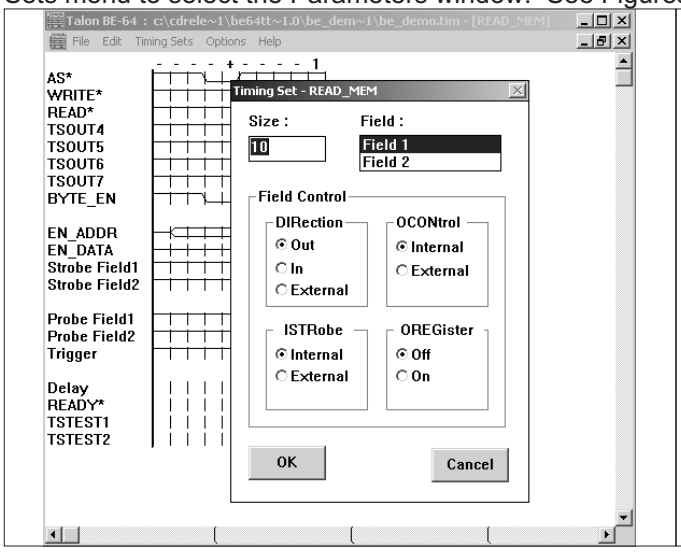


Figure 4-2 READ_MEM Field 1 Control

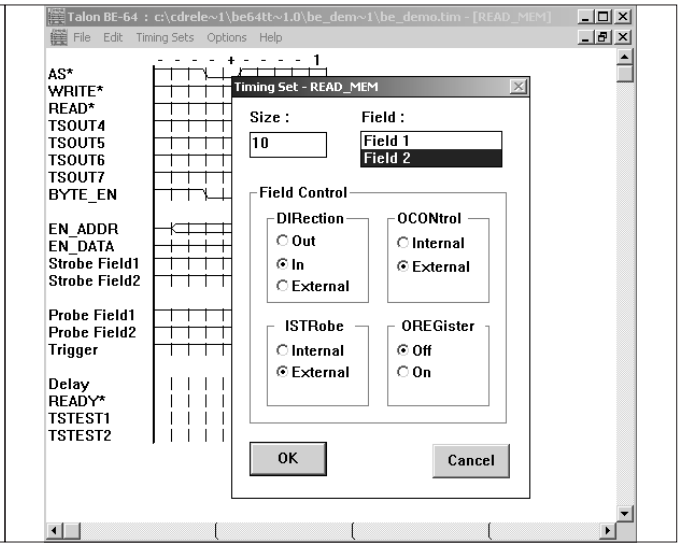


Figure 4-4 READ_MEM Field 2 Control

The next step is to program the actions of the signals used by the Demo Board. For example AS* should be set to LOW in clock cycles 2 and 3 as depicted in the figure below. Complete the setup for each of the required control signals.

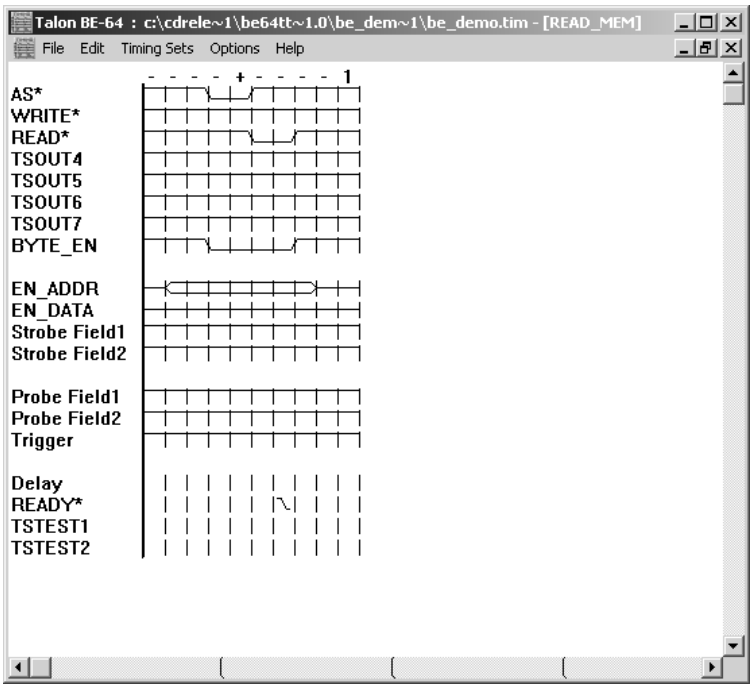


Figure 4-3 Signal Setup READ_MEM Cycle

Now program the WRITE_MEM cycle by repeating the Parameter and signal setup routines in the same manner as for the READ_MEM cycle.

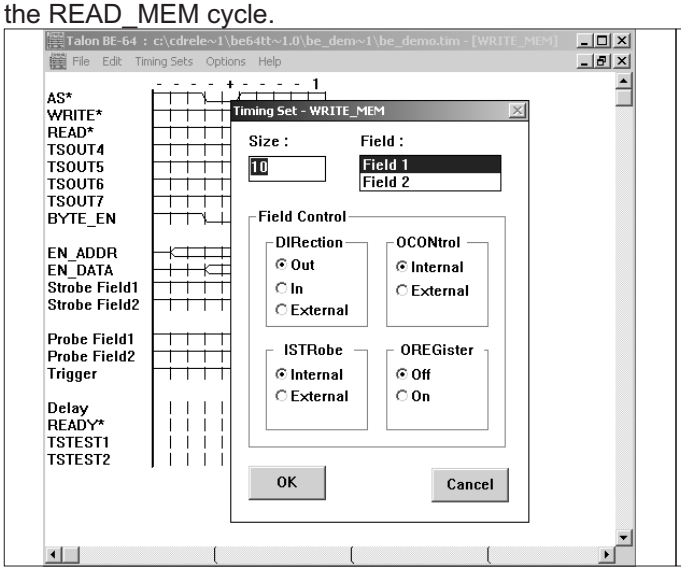


Figure 4-7 WRITE_MEM Field 1 Control

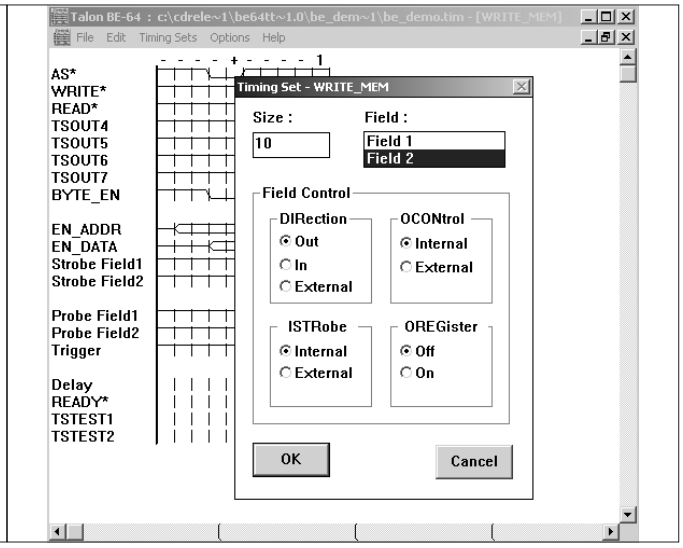


Figure 4-6 WRITE_MEM Field 2 Control

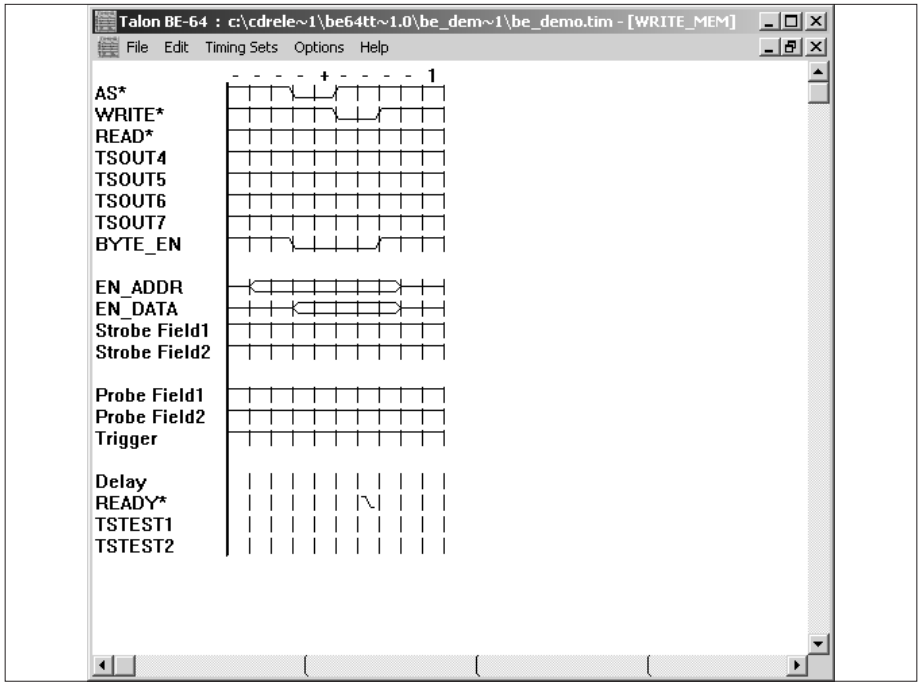


Figure 4-5 Signal Setup WRITE_MEM Cycle

The file should be saved as BE_DEMO.tim and exported as BE_DEMO.tsc using the commands from the File menu. The exported file may be written to the bus emulator card using the BE-64 VXIplug&play drivers or the Soft Front Panel as will be shown in Section 5.,

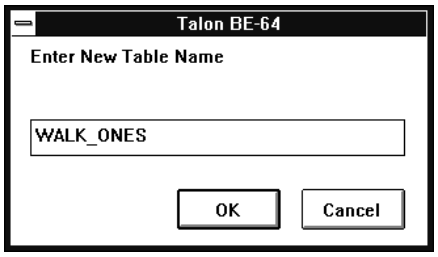
Close the Timing Editor before proceeding.

4.1.2 PROGRAMMING THE FIELD MEMORY

In order to read or write data from the I/O field memory to the demo board, we need to define a table(s) and allocate space for the data. The Table Editor routines may be used to name tables, define length and width and insert data if desired. We will only define one table for use with the demo board. The name of the table is WALK_ONES and we'll fill it with a walking ones pattern, 16 words deep and 16 bits wide. You may program other tables if you wish.

Start the Table Editor program by clicking on the "BE-64 Table Editor", (see right) icon. Click on the Tables menu and select the New command. Enter the name WALK_ONES and click OK.

BE-64 Table Editor



The table may be edited to rename signal mnemonics for documentation by using the Setup command found in the Tables menu. At this time we'll go straight to the Parameters command from the Tables menu to define the field size and characteristics. Field 1 is the address field and will contain the address of the registers on the Demo Board. Field 2 will be the data field and contain the walking ones data pattern.

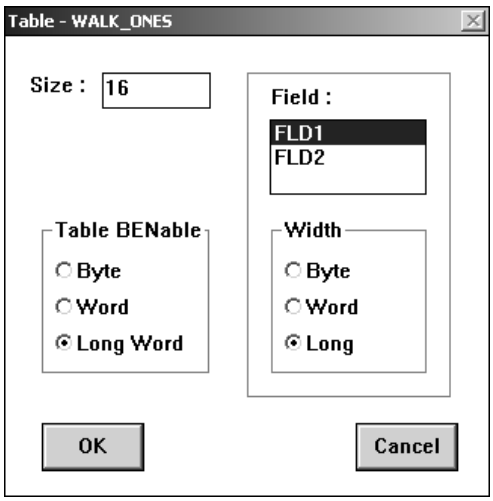


Figure 4-8 Field 1 Parameters

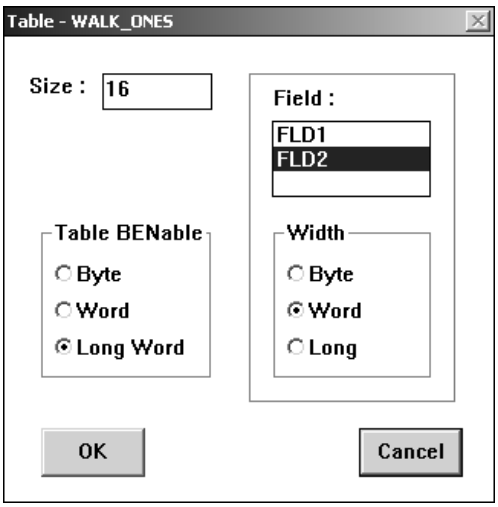


Figure 4-9 Field 2 Parameters

Click on the Tables menu and select the Parameters command. Field 1 will use the Longword format which provides 32 bit wide transfers. Field 2 will be set to 16 bit data (Word). The Size command refers to the number of data words to be stored in this table, (range is 1-64k). In this case we have chosen to define a table containing 16 data words.

In order to fill FLD2 with the walking ones pattern, we need to scroll down to CH 0 of FLD2. Placing the cursor on the first bit of CH 0 and clicking will set the first transfer to 1. Use the cursor to highlight the area to program by going to the channel name area left of the data area and dragging the cursor from CH 0 to CH 15. This will define the memory space to be used by the automatic fill function of the Edit menu.

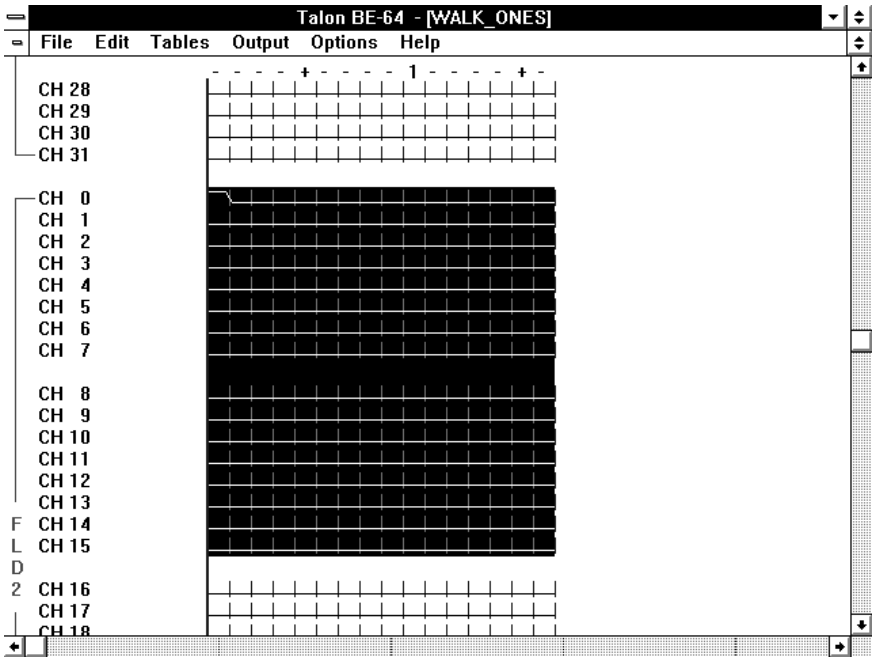


Figure 4-10 Selecting Data Fill Area

Use the Edit menu to select the Fill command. Select the Rotate function and enter a 1 as the rotate value. Clicking on OK will complete the operation. The table should resemble Figure 4.11.

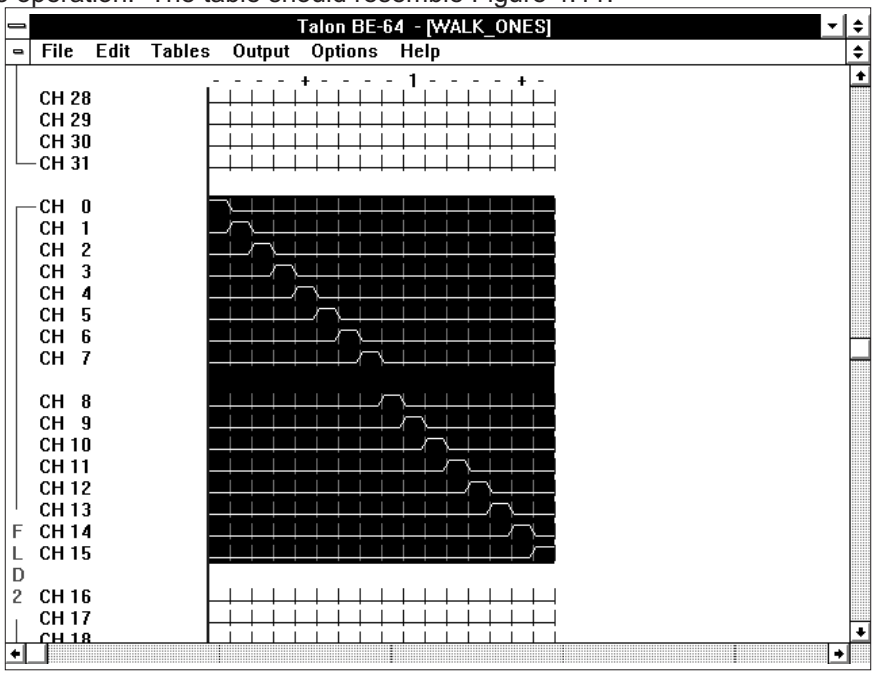


Figure 4-11 Field 2 Walking Ones Data

Entering address data is usually easier in hexidecimal mode rather than graphic. To change the screen format choose the Options menu , select the Display Format command and click on Hex. The address for the registers on the demo board 40000 must be entered into the first transfer of the Field 1 data area. To do this click on the first transfer position in column two. Enter 04 in the pop up data entry window and click on OK. Next, position the cursor above the data field column, (2), to be changed in the field name area. Click to highlight the area. Now use the Edit menu, Fill and Repeat commands to set all the data words to the register address, H040000.

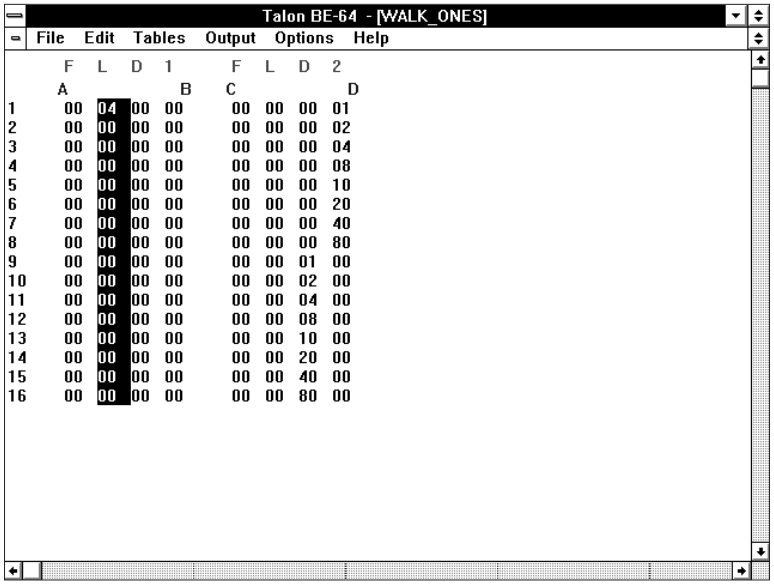


Figure 4-12 Initializing The First Address Data Word

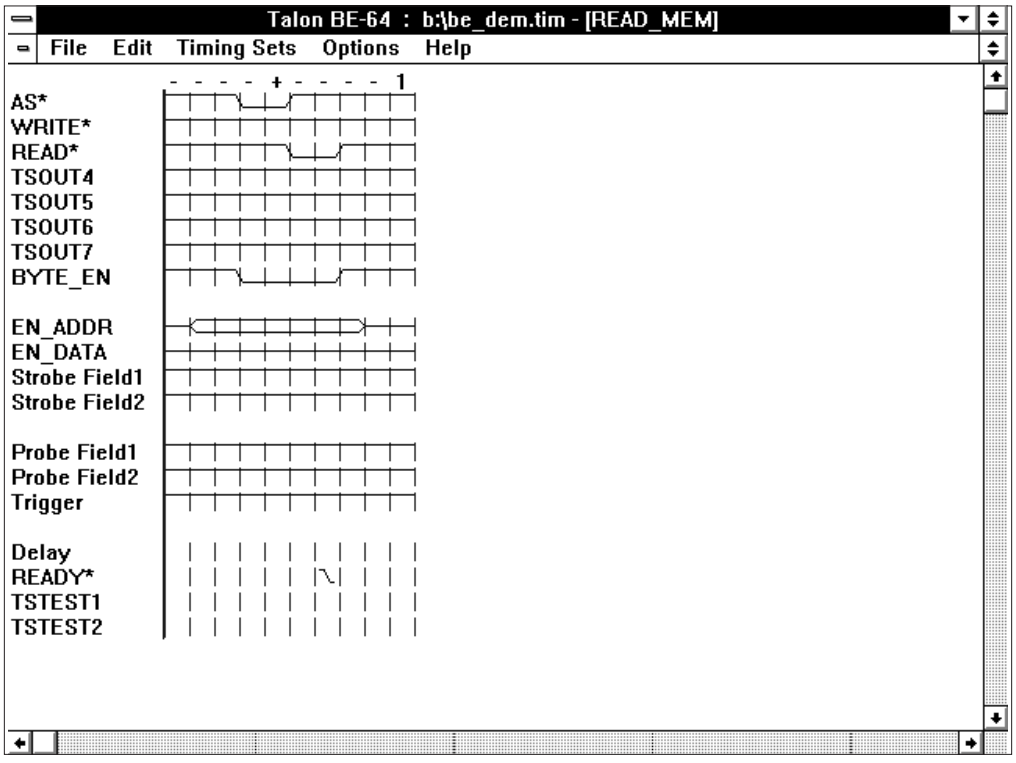


Figure 4-13 Timing Signals for READ_MEM Cycle

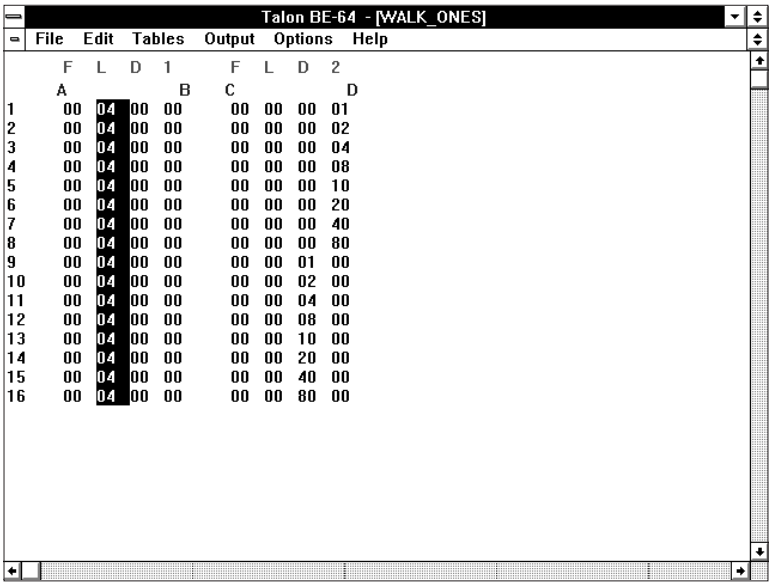


Figure 4-14 Rpeat Fill of Address Data

The file should be saved as DEMO_TAB.tbl and exported DEMO_TAB.tbc using the respective commands found under the File menu. This file is the one we will be downloading to the Bus Emulator for use in Section 5, Demo Board Execution Exercises.

5 DEMO BOARD EXECUTION EXERCISES (Soft Front Panel)

The exercises in this section will use the Soft Front Panel, (SFP,) of the BE-64 VXIplug&play Drivers to perform the demo board exercises. These exercises will range from simple READ and WRITE functions to complete test algorithms such as RAM test.

In this section of the demo procedure the user instructions will be identified as:

SETUP:—demo board setup instructions will follow

ENTER:—input data or word serial command strings will follow

RESPONSE:—expected responses will be displayed on VXI controller monitor

RESULTS:—results to be viewed on the demo board will follow

MANUAL:—interaction with the demo board must be performed

WINDOW:—an inactive window must be opened

COMMAND:—a command should be entered or command button pressed

5.1 DEMO BOARD SETUP PROCEDURES

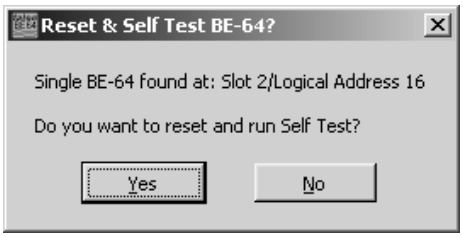
The Demo Board should be connected to the BE-64 via the 40 pin ribbon cables sent with the board. Connectors J10 and J14 of the demo board should be connected to J1 of the BE-64. Connectors J15 and J19 should be connected to J2. The power cables should be connected to a +5V DC, 2.5 amp power supply. Successful power up should illuminate LED D9 next to the power connector. Set the clock to 50 MHz using switch S2.

The BE-64 Bus Emulator module should be installed in the VXI chassis, power on and communication established by clicking on the BE-64 SFP icon (see right). The System should display the initial SFP screen, see Figure 5-1(logical address and slot number may differ).

RESPONSE: Initialization window should indicate the BE-64 has been located and identified by the logical address and slot location.

COMMAND: Click on Yes to select self-test.

RESPONSE: Self-test and Active buttons should display green indicating operational.



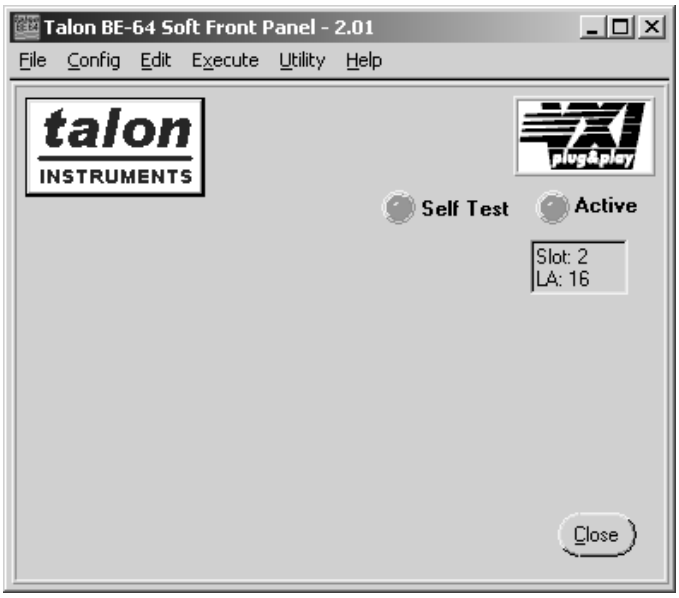


Figure 5-1 Soft Front Panel

The timing sets and data table files created in Section 4 must be downloaded to the BE-64. Perform the following to download the Setup files:

WINDOW: Click on the File menu and select Load Editor File command.

COMMAND: Select the directory containing the *tsc and *.tbc data files.

RESPONSE: The directory will display available *.tsc files.

COMMAND: Select the file BE_DEMO.TSC and click on Load.

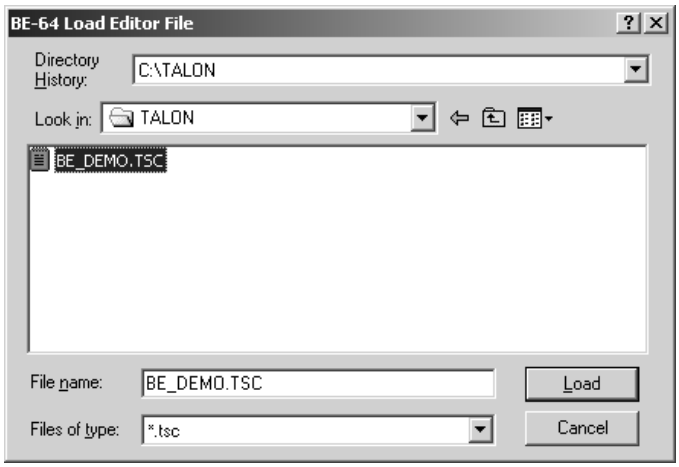


Figure 5-2 Load Timing Data

COMMAND: Click on Yes in response to the query to delete existing timing sets.

RESPONSE: File uploading window will display along with status.

COMMAND: Click on File menu and select Load Editor File command.

COMMAND: Select *.tbc under Files of Type to display the available table files.

RESPONSE: The BE-64 directory will display available *.tbc files.

COMMAND: Select the file BE_DEMO.tbc and click on Load.

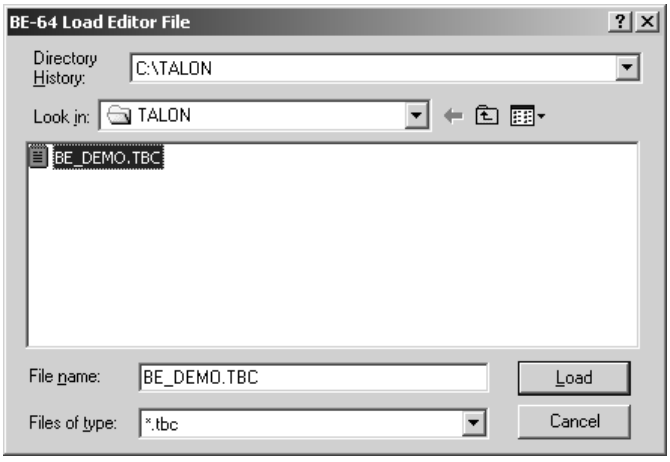


Figure 5-3 Load Table Data

COMMAND: Click on Yes in response to the query to delete existing tables.

RESPONSE: File upload status will display

5.2 EXECUTING TESTS

In the exercises illustrated in this manual we will be using the SFP Execute menu commands to start and stop the individual tests. To activate the Dynamic Execute window perform the following:

COMMAND: Click on the Execute menu and select Dynamic or press F2 on the keyboard.

RESPONSE: The Dynamic execute window will be displayed along with the names of the uploaded timing sets and data tables in their respective pull-down windows.

5.3 READ/WRITE SINGLE DATA TRANSFERS

Simple reads and writes of data to an address are accomplished using the Timing/Data selection under Execution Type, (Timing/Data executes a timing cycle using data directly input/output to FLD1 and FLD2 rather than using data tables stored in the Field Memory RAM). To illustrate using this function we'll output FLD2 data to the Read/Write register at address hex 40000 and input data from the Read Register dip switch at hex 80000 to FLD2. In both cases FLD1 will contain the address data, hex 40000 and hex 80000, to read and write the FLD 2 data.

COMMAND: Select Timing/Data as the Execution Type by clicking on the function or dragging the slide selector.

COMMAND: Select the WRITE_MEM timing set in the Timing Set window selection.

ENTER: Type 40000 in FLD1 entry window as the address data and 5555 in FLD2 window as the pattern data.

COMMAND: Select WORD, (16bits), in the Size selection window.

COMMAND: Click on Run.

RESULTS: The binary data may be viewed on the LEDs located on the demo board. The Register and Data bus LEDs should have D0,D2,D4,D6,D8,D10,D12 and D14 illuminated (hex 5555). The Address bus LEDs should have A18 illuminated (hex 40000).

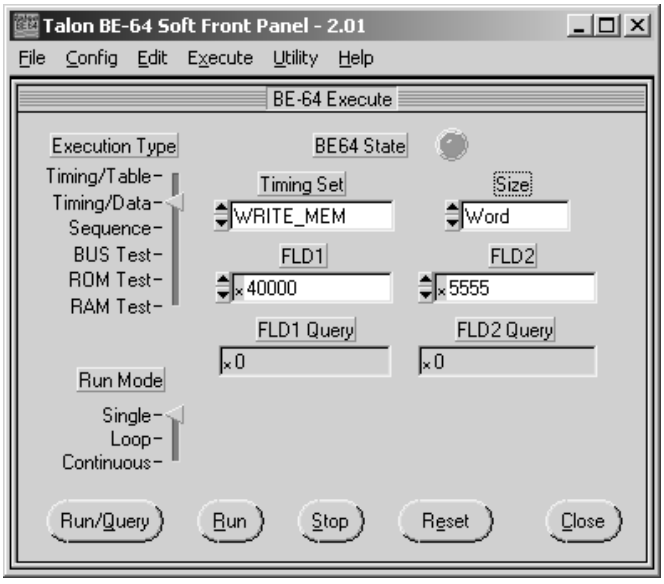


Figure 5-4 Execute Write Register

In order to upload data read from the UUT we must execute the timing cycle with a query command. The Run/Query button operates with Execution Types Timing/Table or Timing/Data, to read and display data from the UUT. In this exercise we'll read the hex data value from the dipswitch in the Read Register Section of the Demo Board, (DIPSW2).

SETUP: Set DIPSW2 in the READ REGISTER section of the demo board with bits D0 and D1 In the up position and switch 9 in the down position. This places a value of hex 3 in the data register and programs the demo board to output the data.

COMMAND: Select the Dynamic window under Execute menu or press F2.

COMMAND: Select Timing/Data as the Execution Type.

COMMAND: Select READ_MEM in the Timing Set window.

COMMAND: Select WORD in the Size window.

ENTER: Enter 80000 in the FLD1 window.

COMMAND: Click on Run/Query.

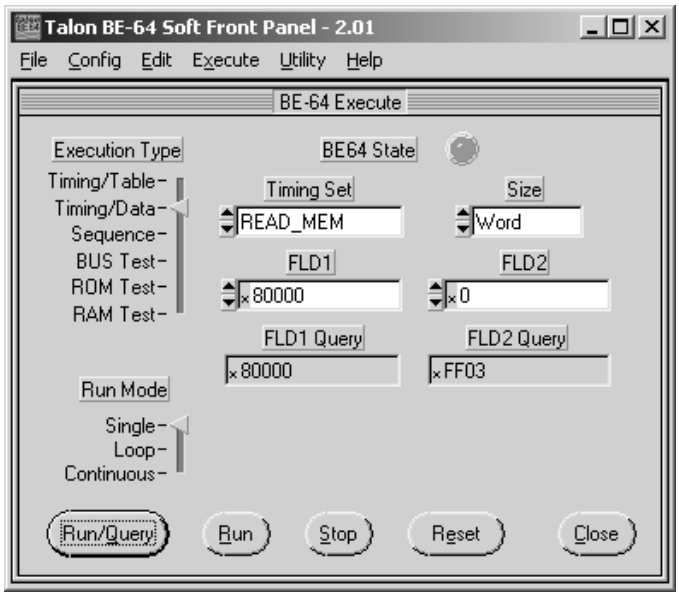


Figure 5-5 Execute Read Register

RESPONSE: The FLD2 Query window will display the value read from DIPSW2, (FF03).

RESULTS: The DATA BUS LEDs should indicate a hex 3 (D0 and D1 illuminated) while the ADDRESS BUS LEDs should display a hex 80000 (A19 illuminated). D8-D15 will also be illuminated.

For more practice using the PIO read function change the settings on DIPSW2 and click on Run/Query to read the new results.

5.4 READ/WRITE DATA TABLES

The reading and writing of tables is similar to reading and writing single data words. The difference is the selected timing cycle will be executed once for each data word in the table. Since the table we have defined has 16 data words there will be 16 cycle executions each time the table is written to or data is read from the UUT.

We'll use the DATA READ/WRITE REGISTER section of the demo board for this exercise. The execution cycles are too fast for viewing (50MHz) as a single pass so we'll use the continuous execution mode to be able to view the results.

COMMAND: Select the Dynamic window from the Execute menu or press F2.

COMMAND: Select Timing/Table as the Execution Type.

COMMAND: Select WRITE_MEM as the Timing Set

COMMAND: Select WALK_ONES as the data table

COMMAND: Select Continuous as the Run Mode

COMMAND: Click on Run

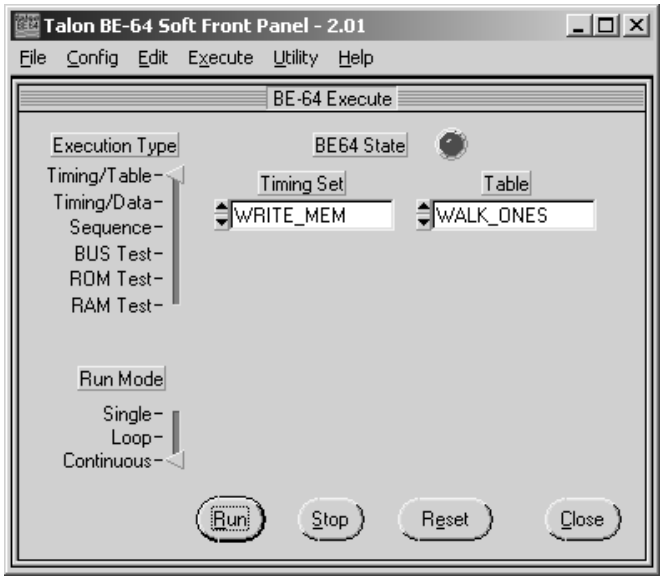


Figure 5-6 Execute Table

RESULTS: The leds in the Data Bus areas will be continuously illuminated. The BUSY and WAIT LEDs will be illuminated on the BE-64.

5.5 HANDSHAKE CONTROL

A unique feature of the BE-64 is it's ability to emulate the handshake or wait state of a typical digital bus. We can demonstrate this feature by using the S3 (STOP HNDSHK) switch on the Demo Board. This switch will inhibit the operation of the READY* signal creating a time-out condition. The time-out occurance will cause the write operation started in Section 5.3 to halt, the BE-64 will illuminate the front panel TIMEOUT LED and a timeout message will be reported via the VXI status register.

MANUAL: Depress the switch S3, STOP HNDSHK on the demo board while the write cycle is executing.

RESULTS: The write operation will halt on the demo board and the TIMEOUT LED on the BE-64 will illuminate.

The handshake control switch S1, STEP HNDSHK can be used to manually toggle the RDY control signal. Since we originally set the timeout value to 1000 clocks we must first disable it to allow enough time for manually activating the READY* signal. The Timing Modules command under the Config menu is used to program the timeout function.

COMMAND: Click the Reset button in the Execute window to halt the current operation.

WINDOW: Open the Timing Modules window under the Config menu.

ENTER: Enter 0 in the Timeout Count window, depress the keyboard Enter key and click on Apply.

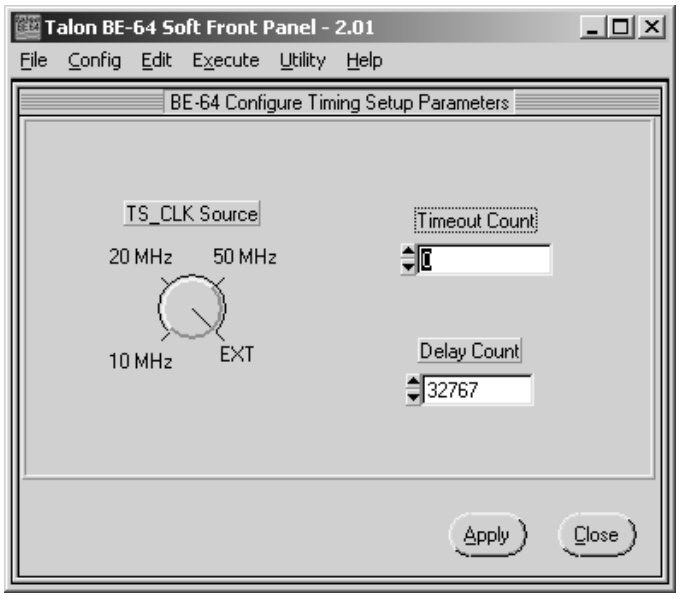


Figure 5-7 Disable Handshake Timeout

WINDOW: Click on Close to return to the Execute window.

COMMAND: Set Execution Type to Timing/Table

COMMAND: Select WRITE_MEM in the Timing Set window

COMMAND: Select WALK_ONES in the Table window

COMMAND: Set Continuous as the Run Mode

COMMAND: Click on Run

RESULTS: Cycle execution will halt with the address bus LEDs illuminated at A18. The data bus LEDs will have only the bit driven by the previous operation illuminated.

MANUAL: Toggle switch S1 on the Demo Board to manually activate the RDY signal.

RESULTS: The data bus LEDs will display a walking pattern for each depression, (handshake), of the S1 switch.

COMMAND: Click on Reset button to halt and clear operation.

5.6 PROGRAMMED I/O

The BE-64 provides 24 bits of static I/O configured as two 8 bit I/O registers and one 8 bit output only register. These I/O signals are separate from the I/O field channels and the timing generator. The PIO INPUT/OUTPUT section of the demo board will be used to illustrate the PIO1 operation.

MANUAL: Set switch 9 of DIPSW1 to OFF (Input position).

WINDOW: Select the Static I/O window from the Config menu.

COMMAND: Select Output in the PIO1 selection area if not already selected.
Click on Apply.

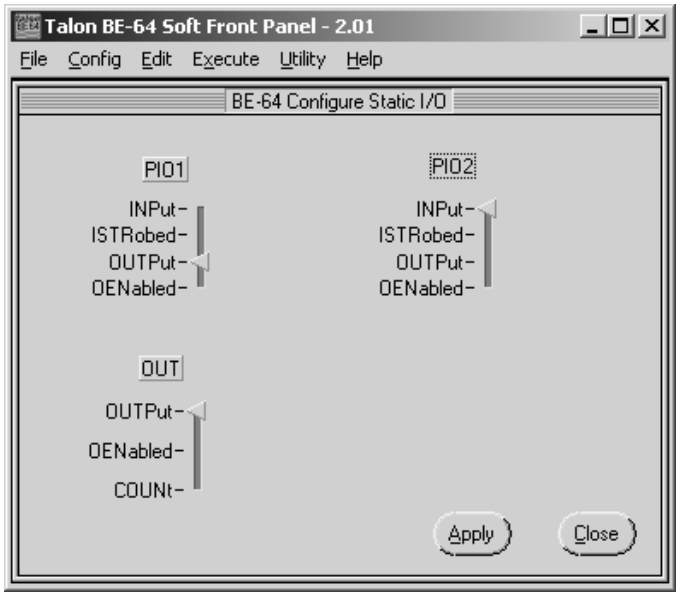


Figure 5-8Set PIO Output

COMMAND: Click on Close.

COMMAND: Select Static command from the Execute menu.

ENTER: Type 55 in the data entry window for PIO1 Data.

COMMAND: Click on PIO1 Output button.

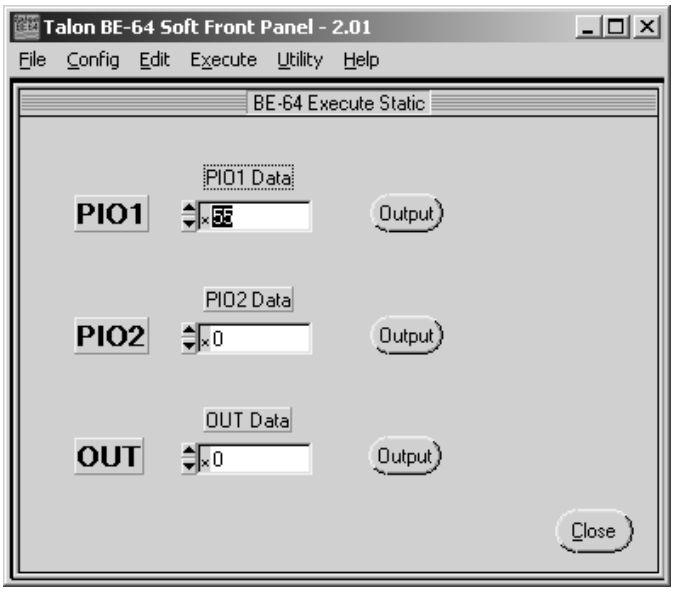


Figure 5-9 Execute PIO Output

RESULTS: The LEDs in the PIO INPUT/OUTPUT section will display the hex 55 by illuminating P1-0,P1-2,P1-4 and P1-6.

ENTER: Type AA in the data entry window

COMMAND: Click on PIO1 Output button.

RESULTS: The LEDs in the PIO INPUT/OUTPUT section will display the new value, hex AA, by illuminating P1-1,P1-3,P1-5 and P1-7.

The PIO section can also be used to illustrate using the PIO1 as an input register.

MANUAL: Set switch 9 on the DIPSW1 to ON (Output position).

WINDOW: Select the Static I/O window under the Config menu.

COMMAND: Select Input under the PIO1 selection area.

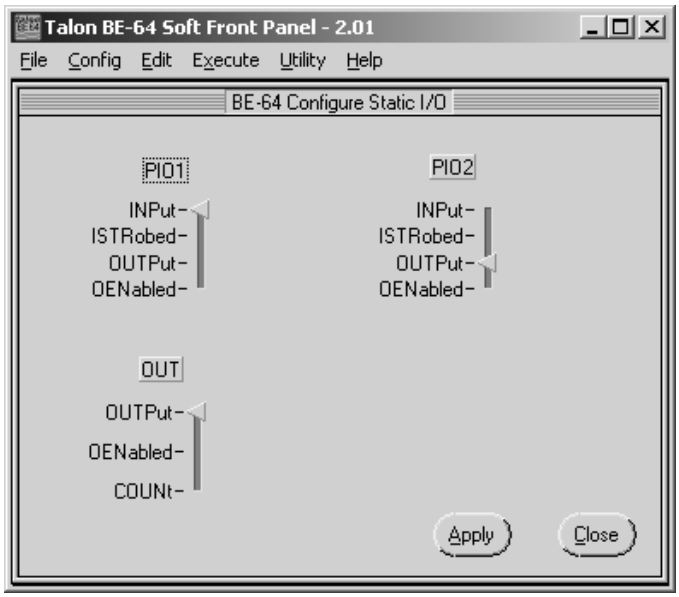


Figure 5-11 Set PIO Input

COMMAND: Click on Apply and Close to return to the Static Execute window.

COMMAND: Click on the PIO1 Output Button, (this will cause the software to read the PIO1 status and reset the function to input).

MANUAL: Program a value on the DIPSW1 to read by setting bits 1 and 3 to the up position.

COMMAND: Click on PIO1 Input button.

RESPONSE: The value returned and shown in the PIO1 Data control will be 5 (binary 00000101).

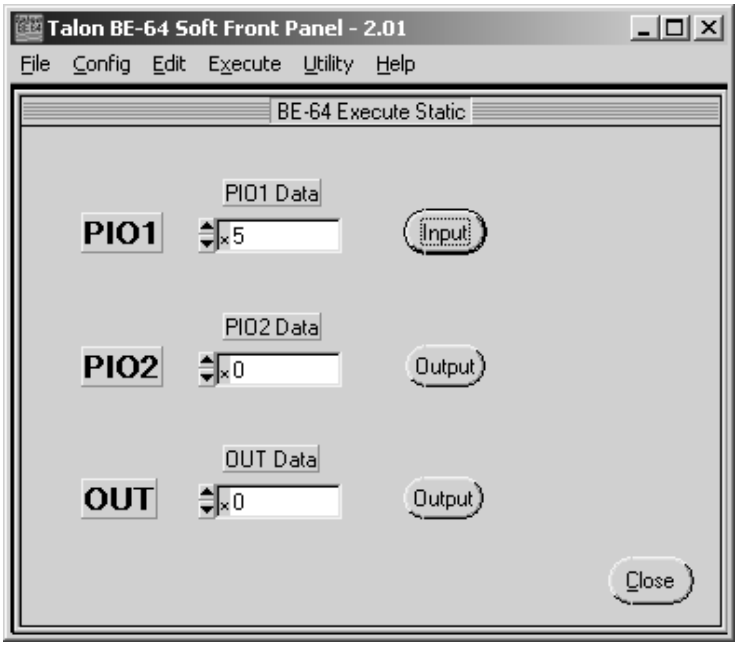


Figure 5-10 Execute PIO Input

RESULTS: The PIO LEDs will be illuminated at P1-0 and P1-2 (hex 5).

5.7 AUTOMATED KERNEL TESTS

The BE-64 has pre-programmed routines for testing kernel logic on the UUT. Kernel testing involves testing three of the most common components on digital controlled boards, RAM, ROM and the digital control bus. We'll demonstrate two of these preprogrammed test functions using the Demo Board.

5.7.1 RAM TEST

The RAM test command will execute the pre-programmed test using FLD1 for the address data and FLD2 for the test pattern data. The RAM test executes WRITE_MEM/READ_MEM timing set cycles at each word address in the range specified by the Start Addr. and Stop Addr. parameters. The Size selection provides the address increment information. The FLD2 Data provides the initial test pattern.

The basic operation of the RAM test is to write the FLD2 Data pattern at each address and read it back for comparison. After a successful pass of the original pattern to all addresses in the range, a second pass will be performed using the complement of the original FLD2 Data pattern.

Any comparison failures will halt the operation and report results to the VXI controller. The results of the test are returned as two numeric values. The first value represents the Fail Address and EXP xor ACTUAL the failing data bits. A value of 0 indicates no failures.

COMMAND: Press Reset button to halt current operation.

MANUAL: End the stop handshake operation by depressing the switch S3 (D7 LED should go off).

WINDOW: Select RAM under Execution Type in the Execute window.

ENTER: Start Addr. hex 400000

ENTER: Stop Addr. hex 407FFF

ENTER: FLD2 Data hex 5555

COMMAND: Select Word in the Size selection window.

COMMAND: Click on Run

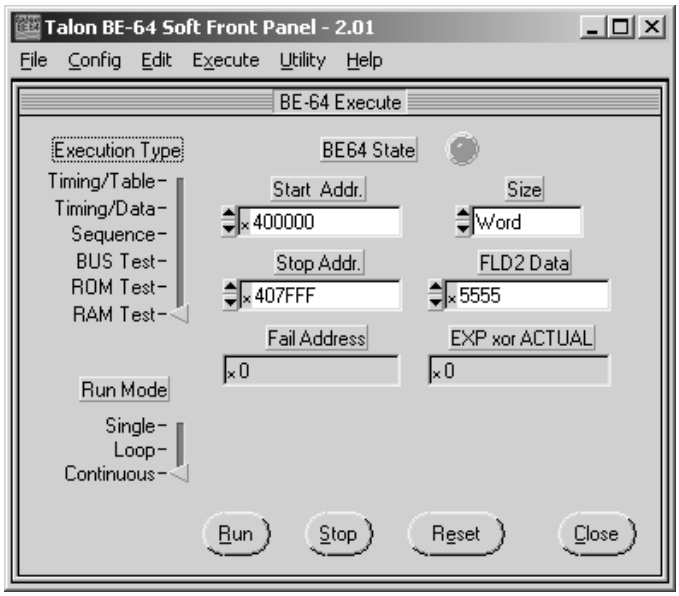


Figure 5-12 Execute RAM Test

RESULTS: The Address Bus LEDs will illuminate in a walking pattern while the test is being performed. Upon completetion the Address LEDs will indicate the final address, (hex 407FFF), while the Data LEDs will show a hex AAAA pattern.

RESPONSE: Passing test will be indicated by 0's in the windows Fail Address and EXP xor ACTUAL.

We'll perform the same test but use the fault insertion section of the demo board to interject faults onto the data and address buses. The first fault will ground data bit D7, resulting in a data bus failure only.

MANUAL: Depress and hold switch S6,(GND D7), in the Fault Insertion section.

COMMAND: Click on Run.

RESPONSE: The test should detect the error, Fail Address will display hex 400000 (indicating the RAM failed at the beginning address) and the EXP xor ACTUAL window will display a hex 80 (binary 1000 0000) indicating bit D7 failed on the data bus. If you would like to see a binary display of the data click on the small x to the left of the data value in the EXP xor ACTUAL window and select Binary from the pull down selector. The bit position with a one will indicate the failing data bit.

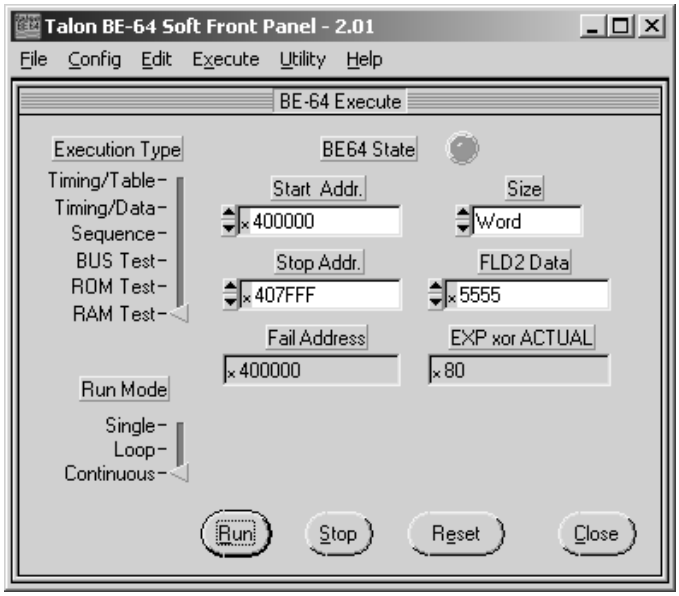


Figure 5-13 Execute RAM Test D7 Error

The second fault we'll insert is with switch S7, RAM ERROR D13/A7, which will tie data bit 14, (D13) to address bit 8, (A7).

MANUAL: Depress and hold switch S7, RAM ERROR D13/A7, in the Fault Insertion section.

COMMAND: Click on Run.

RESPONSE: The results shown this time will show the test failed at address A7, (hex 400080), and data bit D13, (hex 2000 or binary 10000000000000).

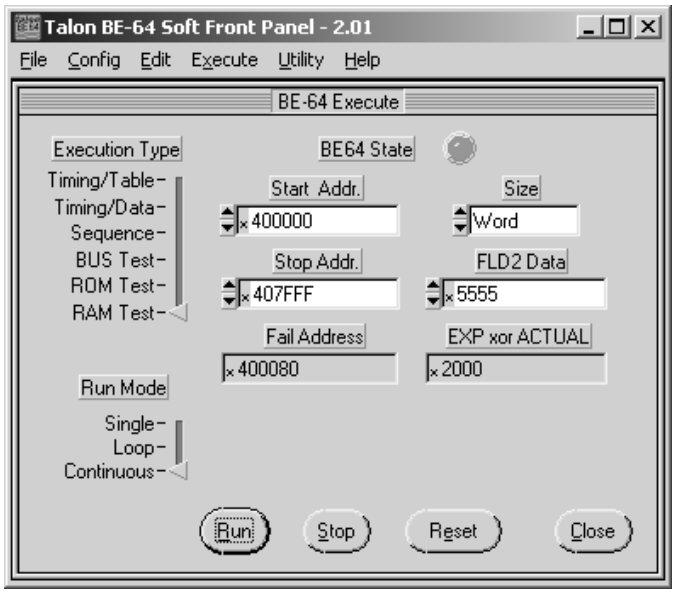


Figure 5-14 Execute RAM Test A7/D13 Error

5.7.2 ROM TEST

The ROM test will execute the pre-programmed test using FLD1 as address data of the ROM. The ROM test executes the READ_MEM timing set sequence at each FLD1 address in the range specified by the start and stop parameters using the Size parameter as the address increment. A 32 bit checksum value will be calculated which can be saved by your test program to compare when testing boards in the future to determine whether data stored in a ROM is correct or not.

WINDOW: Select ROM as the Execution Type in the Execute window.

ENTER: Perform the following entries:

Start Addr. hex 00

Stop Addr. hex FFFF

COMMAND: Select Word in the Byte Enable selection window.

COMMAND: Click on Run.

RESULTS: The address and data bus LEDs will illuminate in a walking pattern while the test is being performed.

RESPONSE: The calculated cecksum will be displayed as #H200F18BA in the Checksum display window.

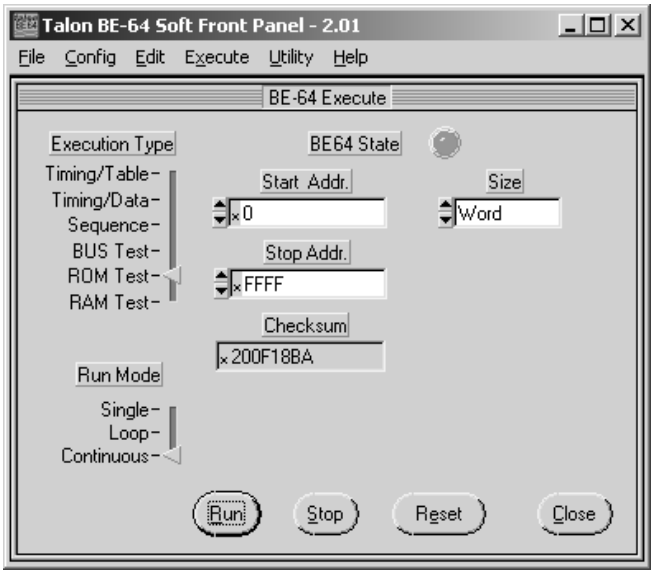


Figure 5-16 Execute ROM Test

We'll use the Fault Insertion switch S6, GND D7, to illustrate the results of a failing data bit.

MANUAL: Depress and hold switch S6, GND D7, in the Fault Insertion section while the test is being performed.

COMMAND: Click on Run.

RESULTS: The address and data bus lLEDs will illuminate while the test is being performed.

RESPONSE: The checksum returned will show a different value from that calculated as the results of our passing test. This indicates either the data stored in the ROM is bad or there is a problem with the buses transfering the data. Other tests will have to be performed to determine the exact cause of the failure.

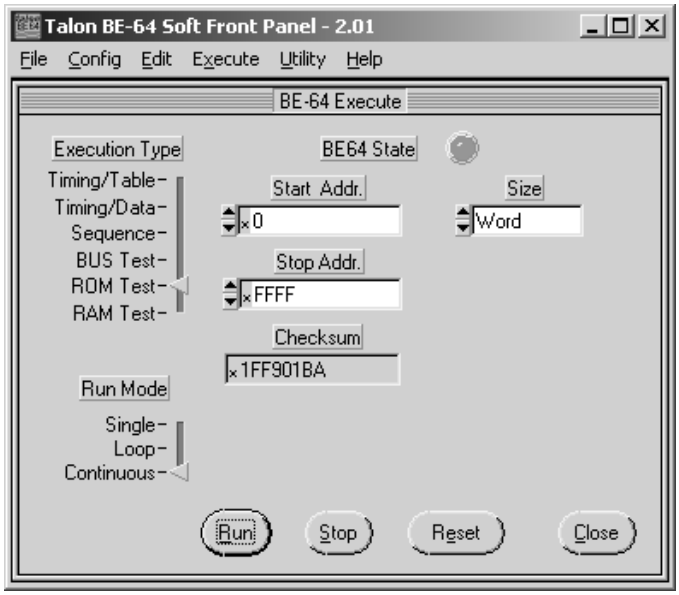


Figure 5-15 Execute ROM Test Error

At this point in the demo board exercises you may proceed to the Advanced Programming Section or develop further exercises of your own.

6 ADVANCED PROGRAMMING TECHNIQUES (Soft Front Panel)

This section is provided to allow the user to exercise some of the more advanced capabilities of the Model BE-64.

The Model BE-64 has a CALCULATE subsystem as part of it's SCPI language. The commands in the CALCULATE subsystem may be used to execute test functions internal to the BE-64 which would normally have to be performed by the VXI controller. Performing these functions in the BE-64 saves significant time by eliminating the need to pass data tables back and forth between the BE-64 and the VXI controller.

6.1 DATA TABLES

We need to create two data tables to perform the exercises in this section. The first table will be used to create a data pattern to write to the demo board RAM. The second table will be used to store the recorded data read from the demo board RAM. The following exercises create two tables RAM_PATT and RAM_READ.

We'll use the SFP Data Table window to create the first table RAM_PATT.

COMMAND: Click on Data Tables under the Edit menu.

COMMAND: Click on New to name and open a table.

ENTER: Type the first table name, RAM_PATT in the entry window.

ENTER: Enter 32 in Number of Words.

COMMAND: Click on Apply.

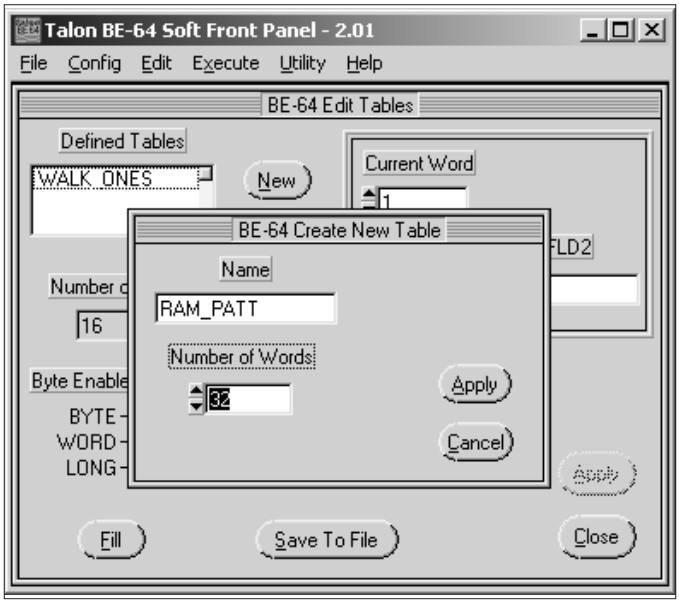


Figure 6-1 Define Table

Now that we've defined the parameters of RAM_PATT we need to load data into the table. To start, we'll use the Fill function to define address data in Field 1. The start data value is the Demo Board RAM address, hex 400000, and we'll increment the addresses by 2 for the remainder of the 32 words. After that, we'll load a rotating ones pattern in the Field 2 data memory.

COMMAND: Click on the Fill button in the Edit Tables window.

COMMAND: Select Increment as the Fill Function.

ENTER: Enter 2 as the Increment Value.

ENTER: Enter 400000 as the 1st Word Value.

COMMAND: Select FLD1 as the Field Select and Long as the Field Width.

COMMAND: Click on Apply.

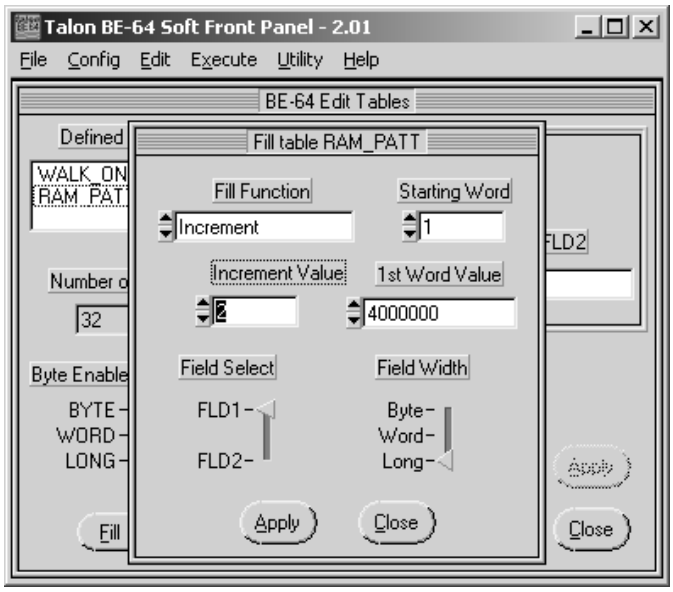


Figure 6-2 Fill Increment

We now have to program Field 2 memory to provide data to be written to the RAM at the addresses we just programmed in Field 1 memory.

COMMAND: Click on Fill.

COMMAND: Select Rotate as the Fill Function.

COMMAND: Leave 1 as the Starting Word and as the Rotate Value.

ENTER: Enter 1 as the 1st Word Value

COMMAND: Select FLD2 as the Field Select and Word as the Field Width.

COMMAND: Click on Apply.

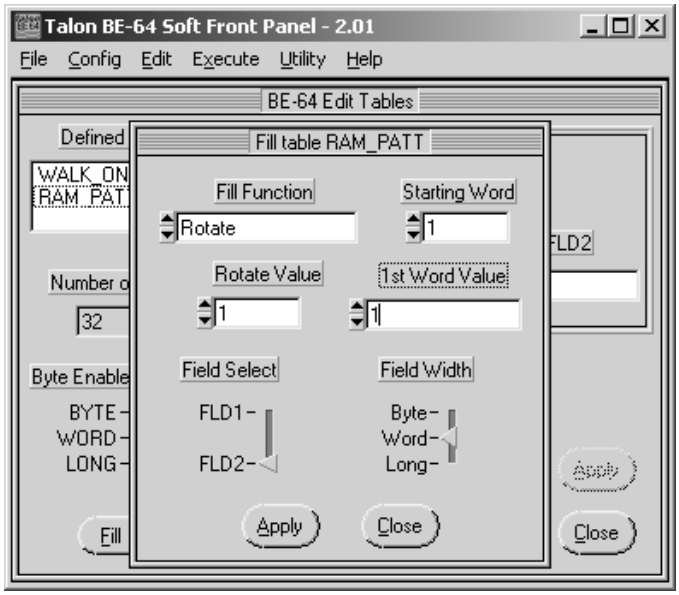


Figure 6-3 Fill Rotate

COMMAND: Click on Close.

NOTE: In the Edit Tables window you may view the data stored in the new table, RAM_PATT, by clicking on the up button beside the Current Word window. By continuously clicking on the up button you may scroll through all the 32 words to verify correct data.

In order to illustrate the table compare function of the BE-64 we need to create a second table to store the data we read from the RAM of the Demo Board. We'll name this table RAM_READ. In the other exercises we used the Table Editor and the SFP to create the tables. We'll use another function of the BE-64 to create the third table. The function we'll use is the Interactive window which allows us to directly execute BE-64 SCPI commands.

COMMAND: Select Interactive from the Utility menu.

ENTER: Enter the command string TABL:DEF RAM_READ,RAM_PATT, (this command creates a new table named RAM_READ identical to RAM_PATT).

ENTER: Depress the Enter key on the keyboard to send the command to the BE-64.

RESPONSE: The Active light will illuminate while the data is being transferred.

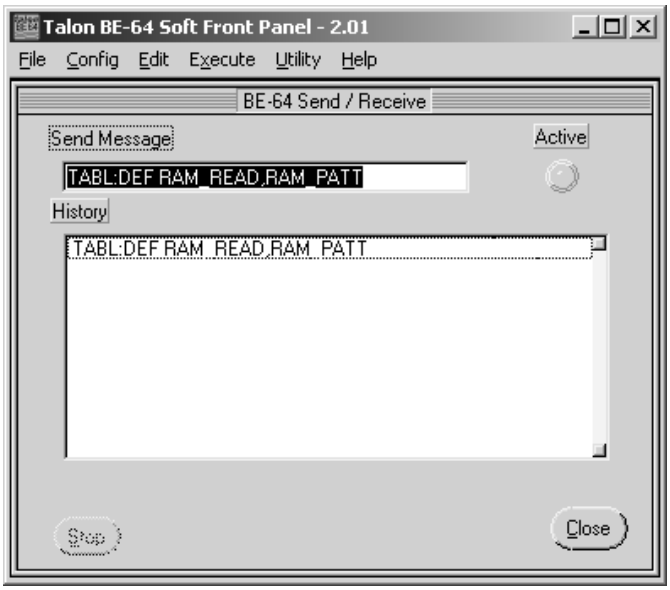


Figure 6-4 Interactive Command

COMMAND: Click on Close.

The second table, RAM_READ may be checked by use of the Edit Table window to ensure the table was created correctly.

COMMAND: Select Data Tables from the Edit menu.

COMMAND: Select RAM_READ as the Table

COMMAND: Using the Current Word up arrow scroll through the data to verify all 32 data words were copied correctly.

6.2 TABLE COMPARE FUNCTION

The first part of this exercise involves writing the table RAM_PATT to the RAM of the demo board. This is easily accomplished by using the Dynamic execution window.

COMMAND: Select the Dynamic window from the Execute menu or press F2.

COMMAND: Select Timing/Table as the Execution Type.

COMMAND: Select WRITE_MEM as the Timing Set.

COMMAND: Select RAM_PATT as the Table.

COMMAND: Select Single as the Run Mode.

COMMAND: Click on Run.

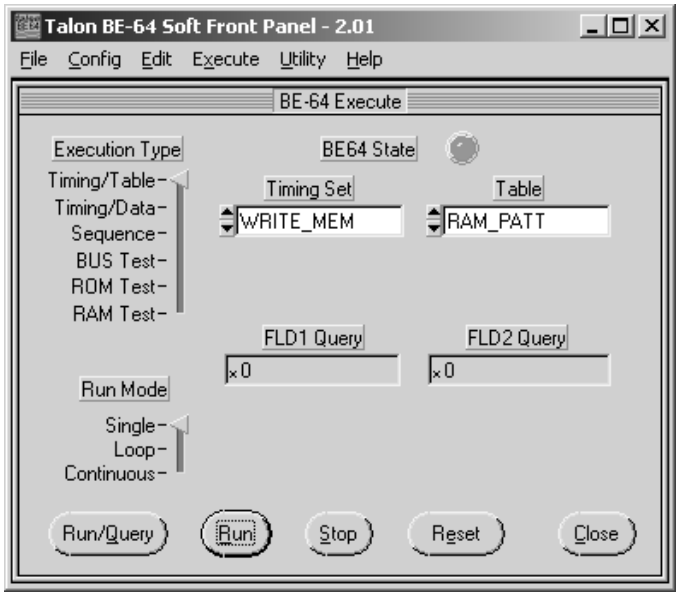


Figure 6-5 Table Compare Write

In order to illustrate the table compare function we must first read and record data into our second table. The following procedure will accomplish reading the data stored in the Demo Board RAM into the table RAM_READ. To illustrate the use of the Table Compare function to detect a problem we'll use one of the fault insertion switches, S6, which grounds data bit D7 to corrupt the data read into the RAM_READ table.

COMMAND: Select READ_MEM as the Timing Set.

COMMAND: Select RAM_READ as the Table.

COMMAND: Select Single as the Run Mode.

MANUAL: Depress and hold switch S6, GND D7 on the Demo Board..

COMMAND: Click on Run.

The newly recorded data overwrote the data in Field 2 of RAM_READ and we can now perform a table compare to find the differences between our original data in table RAM_PATT versus the data now stored in table RAM_READ.

COMMAND: Select the Execution Results from the Utility menu.

COMMAND: Select RAM_PATT as Table 1.

COMMAND: Select RAM_READ as Table 2.

COMMAND: Select FLD2 as Field, (field 2 is the only one we recorded data into).

COMMAND: Click on Table 1 / 2 Compare button.

RESPONSE: The following responses will be shown:

No. of Failures: 2

Failing Channels: hex 80 (indicating bit D7)

First Failure Word: 8

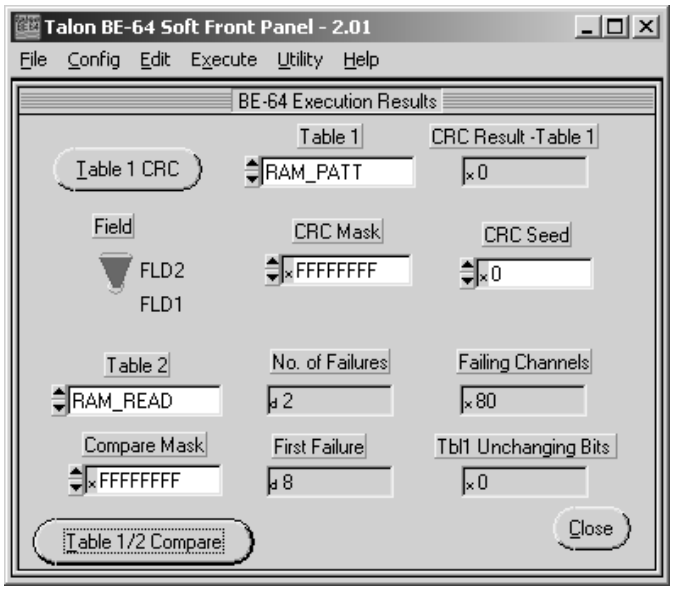


Figure 6-6 Table Compare Results

6.3 CRC CALCULATION

The BE-64 will execute a 16 bit polynomial CRC on a specified table in the field memories. The CRC algorithm provides the capability to include a SEED and MASK value to extend the test functionality. The seed can be used to accumulate several CRC's into one. The mask function can be used to detect which channel failed in case of stuck or shorted channels.

Since we have two tables , one of which is our correct data and one of which has corrupted data we can use them to calculate the CRC's and view the difference.

The first step is to calculate the "good" CRC from RAM_PATT table.

COMMAND: Select the Execution Results window from the Utility menu if not already selected.

COMMAND: Select RAM_PATT as Table 1.

COMMAND: Select FLD2 as the Field.

COMMAND: Click on the Table 1 CRC button.

RESPONSE: The data in the CRC Result-Table is 791E.

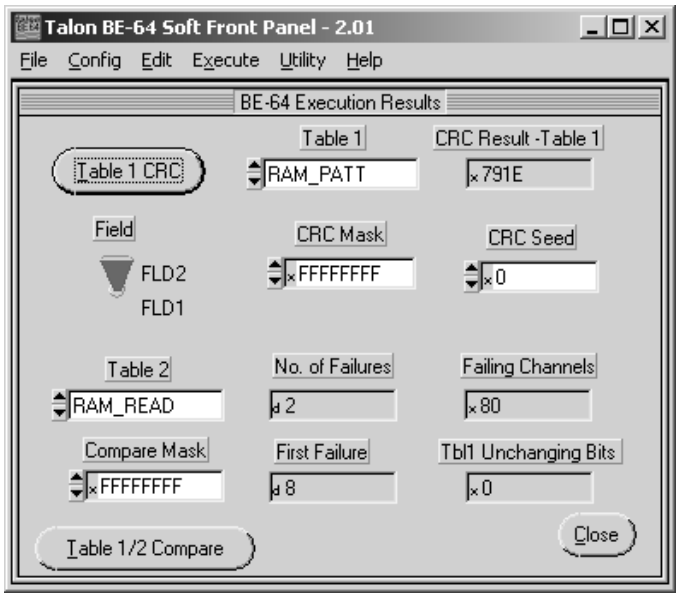


Figure 6-8 Table CRC Pass

The CRC from RAM_READ should be different since the table compare showed data that didn't match.

COMMAND: Select RAM_READ as Table 1.

COMMAND: Select FLD2 as the Field.

COMMAND: Click on the Table 1 CRC button.

RESPONSE: The data in the CRC Result-Table 1 window is CE0 indicating a failure.

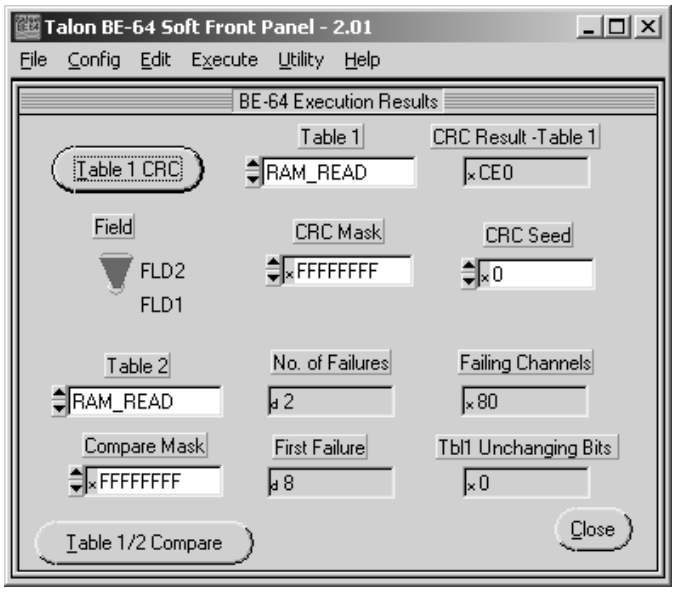


Figure 6-7 Table CRC Fail

6.4 BYTE ADDRESSING

Most bus and microprocessor structured interfaces have separate control lines to indicate the present size of the data bus (byte, word, or longword). For example, the VME bus has "LONGWORD,DS0,DS1" while the 80486 has BE0-BE3. In addition byte and word transfers may be defined to be in the LSB or MSB positions.

The byte enable output signals of the BE-64 are defined by respective VXI compatible commands. The active condition of the BE0- through BE3- is defined by the present transfer type (byte,word or longword), as well as bits 0 and 1 of Field 1 (which represents A0 and A1 of the address bus). Timing set signal #8 (TSOUTT8) is used to internally enable BE0-BE3.

The demo board was designed to operate as a 16 bit bus or by inserting a jumper at the BYTE ADDRESSING pins switch the board to a byte oriented operation. The following block diagram illustrates the demo board data bus architecture.

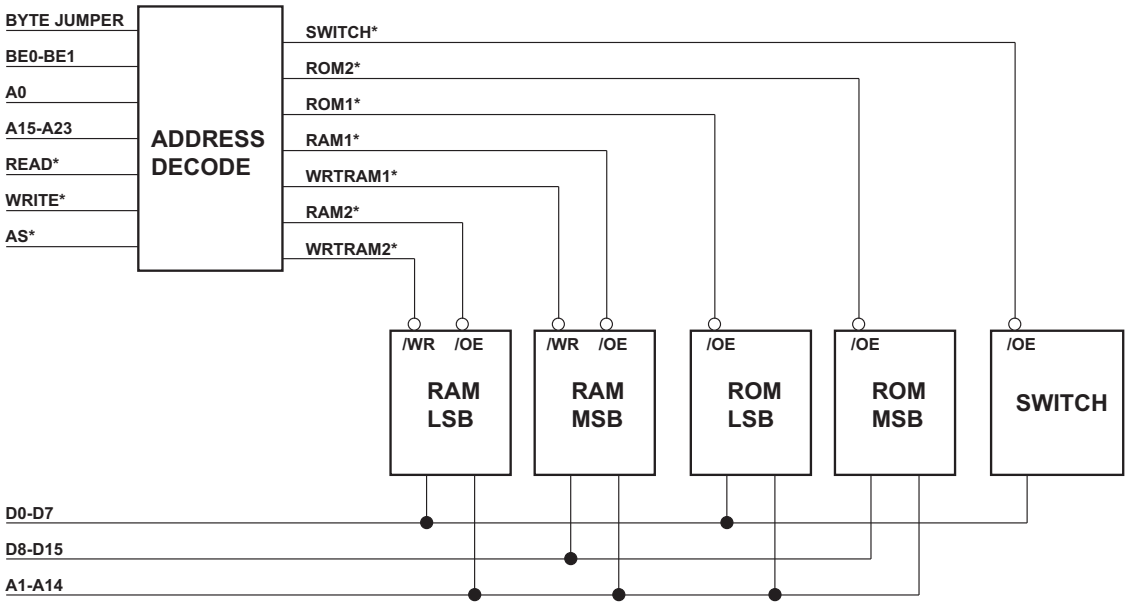


Figure 6-9 Demo Board Bus Block Diagram

Without the BYTE ADDRESSING jumper installed the BE0* and BE1* will have no effect on the data bus operation. The write and read cycles to the RAM will have valid 16 bit data regardless of the action of the BE0* and BE1* signals as shown in the following block diagram.

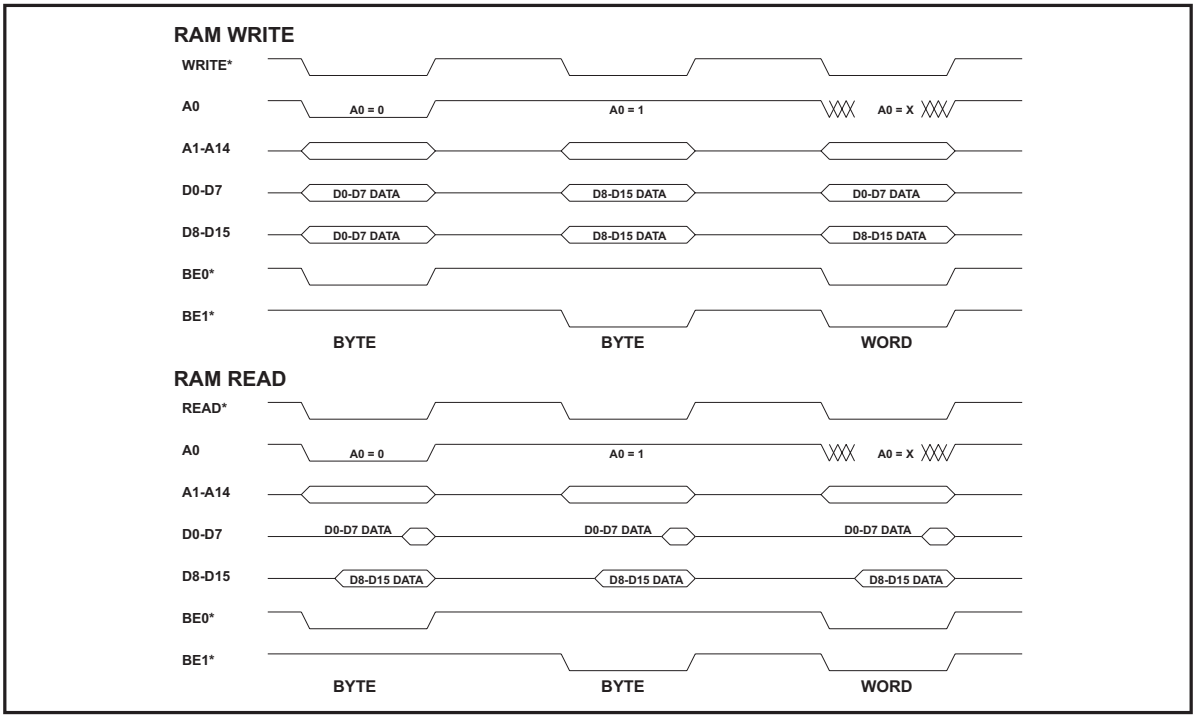


Figure 6-10 ByteEnable Cycles Without Jumper

With the BYTE ADDRESSING jumper installed the data bits D0-D7 and D8-D15 are controlled by the BE0* and BE1* signals. During BYTE operations BE0* controls D0-D7 and BE1* controls D8-D15. During WORD operations both signals must be active to execute a 16 bit word. The following block diagram illustrates this architecture.

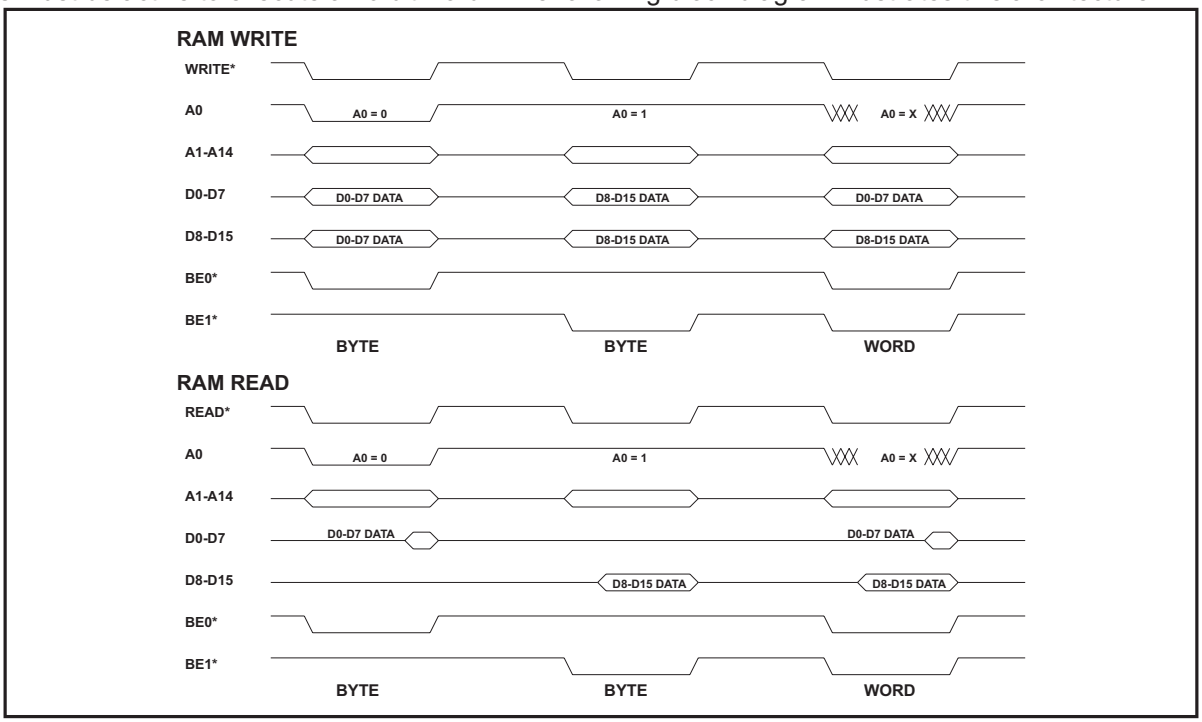


Figure 6-11 Byte Enable Cycles With Jumper

The first exercise illustrates the operation of the normal 16 bit bus. We'll write a 16 bit word at RAM address hex 400000 and then attempt to read 8 bit byte data back at addresses hex 400000 and 400001.

COMMAND: Select Dynamic from the Execute menu.

COMMAND: Select Timing/Data as the Execution Type.

COMMAND: Select Word for Size.

COMMAND: Select WRITE_MEM as Timing Set.

ENTER: Enter #H400000 in FLD1 and #H1234 in FLD2.

COMMAND: Click on Run.

RESULTS: Address Bus LED A22 and Data Bus LEDs D2,4,5,9,12 will be illuminated.

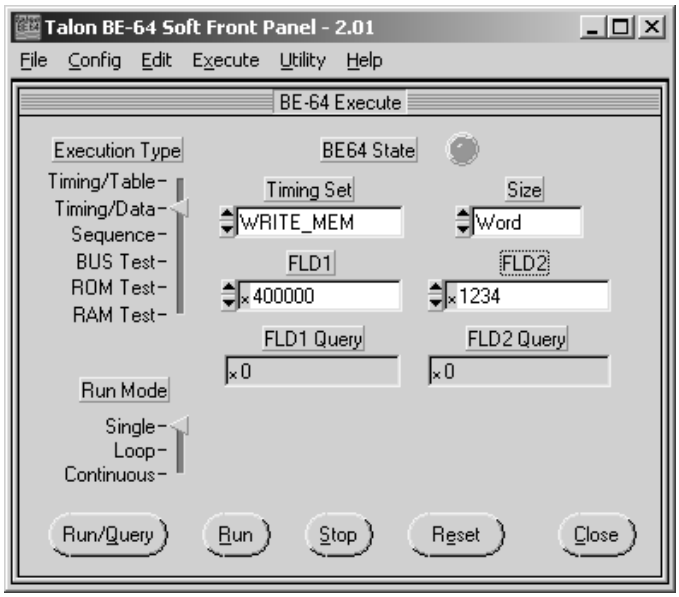


Figure 6-12 Byte Enable Exercise 1

COMMAND: Select READ_MEM as Timing Set,

COMMAND: Click on Run/Query.

RESPONSE: In FLD2 Query data returned #H1234

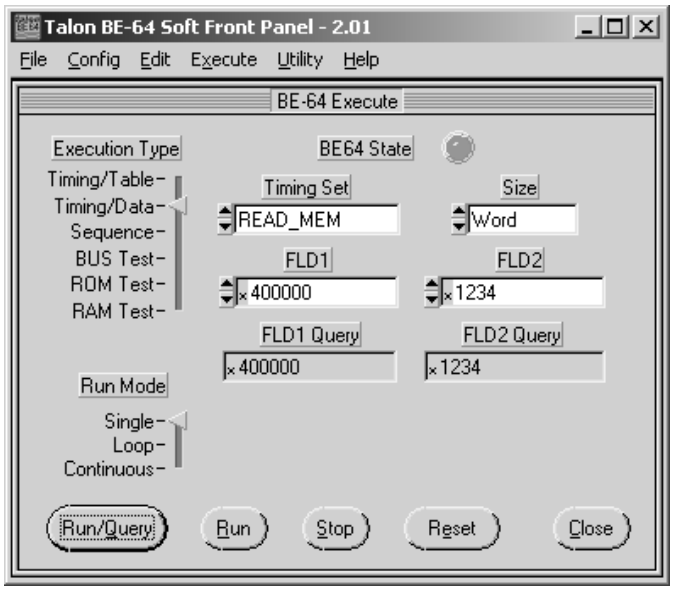


Figure 6-14 Byte Enable Exercise 2

Now we'll try to read Byte size data rather than Word size data. The lower data byte should contain H34 and the upper H12. We'll read the lower byte first at address H400000 followed by the upper byte at address H400001.

COMMAND: Select Byte for Size,

ENTER: #H400000 as FLD1 address.

COMMAND: Click on Run/Query .

RESPONSE: In FLD2 Query window data returned #H34

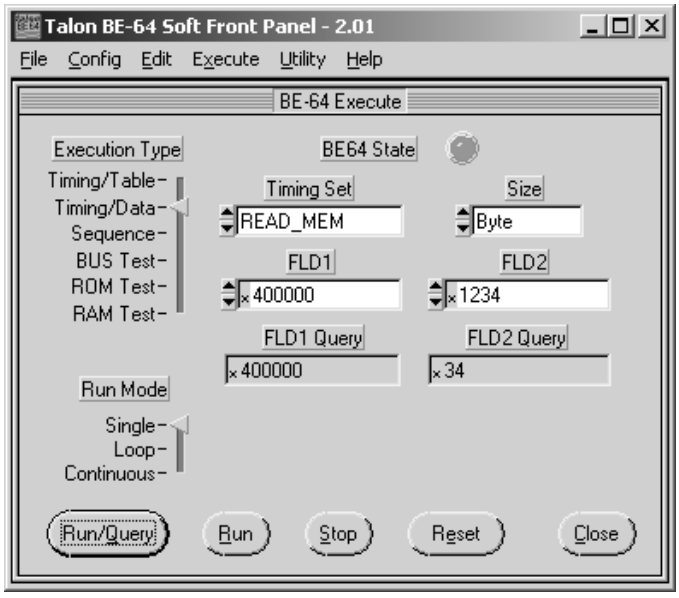


Figure 6-13 Byte Enable Exercise 3

ENTER: #H400001 as FLD1 address.

COMMAND: Click on Run/Query.

RESPONSE: In FLD2 Query window data returned #H34

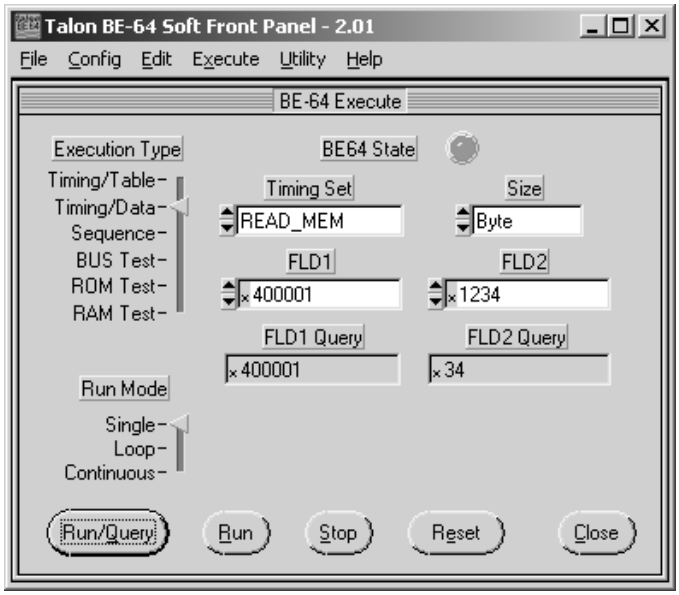


Figure 6-15 Byte Enable Execise 4

As shown in the above exercises it is not possible to read or write data to odd size addresses when using a 16 bit data bus. Data is not accessable on 8 bit boundaries. This creates a problem in digital design since some components on a board may be addressed on 8 bit boundaries while others are addressed on 16 or 32 bit boundaries. The BE-64 allows the user to program the byte enable codes on a word by word or table by table basis to be able to address 8, 16 or 32 bit boundaries.

In this exercise we'll change the demo board addressing scheme to that of a typical byte/word architecture and program the byte enable control signals of the BE-64, (if you are not familiar with the byte enable architecture of the BE-64 please review the BE-64 Operation Manual).

In the next exercises we'll use the Byte Enable command under the Config menu to set the byte enable control signals of the BE-64. We must also change the jumper to switch the addressing architecture of the Demo Board to that of an 8 bit byte enabled architecture as described earlier.

MANUAL: Install BYTE ADDRESSING jumper on jumpers labeled BYTE on the Demo Board.

The following commands will program the byte enable control signals and turn the byte enable function on.

COMMAND: Select Byte Enable from the Config menu.

COMMAND: Select Byte for Byte Enable Code

ENTER: Hex 11 in Byte 1
Hex 22 in Byte 2
Hex 11 in Byte 3
Hex 22 in Byte 4

COMMAND: Set State to ON

COMMAND: Click on Apply

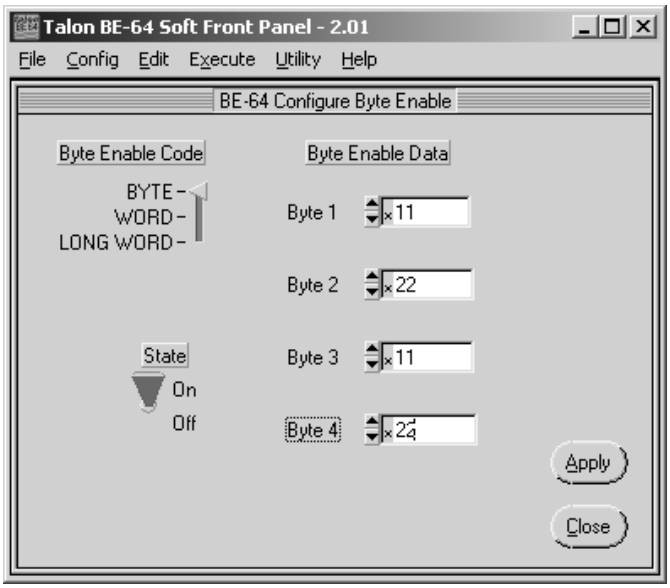


Figure 6-17 Setting Byte Enables for BYTEs

COMMAND: Select Word for Byte Enable Code

ENTER: Hex 33 in Byte 1
Hex 33 in Byte 2
Hex 33 in Byte 3
Hex 33 in Byte 4

COMMAND: Set State to ON

COMMAND: Click on Apply

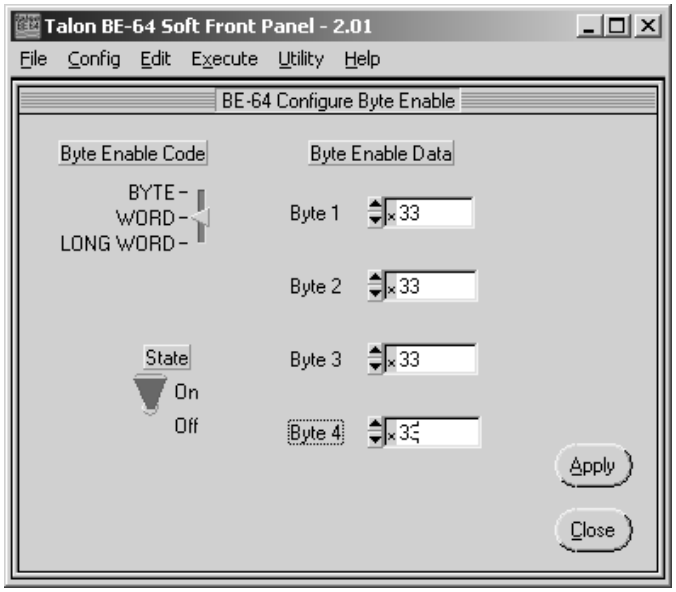


Figure 6-16 Setting Byte Enable for WORDs

We'll use the same commands as before to illustrate the ability to address the RAM in either byte or word format.

COMMAND: Select Dynamic from the Execute menu.

COMMAND: Select Timing/Data as the Execution Type.

COMMAND: Select Word for Size.

COMMAND: Select WRITE_MEM as Timing Set.

ENTER: Enter #H400000 in FLD1 and #H1234 in FLD2.

COMMAND: Click on Run.

RESULTS: Address Bus LED A22 and Data Bus LEDs D2,4,5,9,12 will be illuminated.

COMMAND: Select READ_MEM as Timing Set,

COMMAND: Click on Run/Query.

RESPONSE: In FLD2 Query data returned #H1234

This exercise illustrated the ability of the BE-64 to address word data with the full 16 bit operation just as we did without the byte enable signals programmed. Now we'll illustrate using the Byte operation to access 8 bit address boundaries, which we were not able to do with 16 bit addressing only.

COMMAND: Select READ_MEM as Timing Set,

COMMAND: Select Byte as the Size.

COMMAND: Click on Run/Query.

RESPONSE: In FLD2 Query data returned #H34

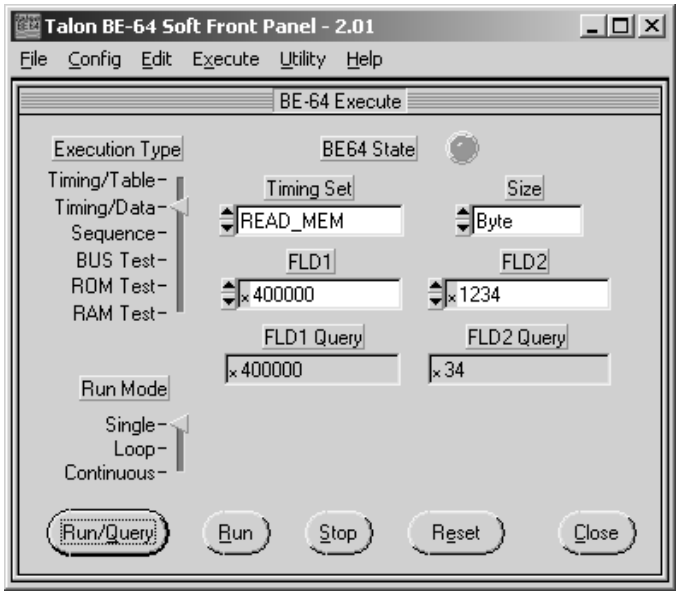


Figure 6-18 Byte Enable Exercise 5

ENTER: Enter 400001 in FLD1.

COMMAND: Click on Run/Query.

RESPONSE: In FLD2 Query data returned #H12.

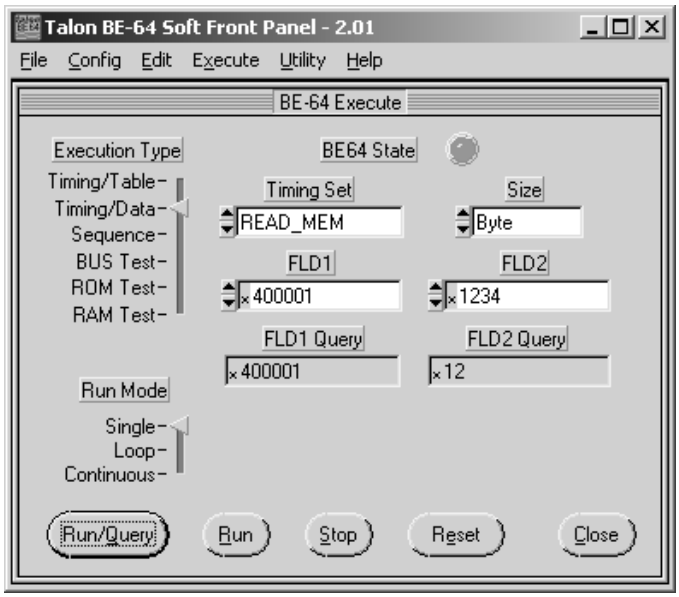


Figure 6-19 Byte Enable Exercise 6

The above examples illustrate the use of the byte enable control function of the BE-64 to emulate a simple byte/word bus architecture. More complex buses,(VME, 80486,60040 etc.) which use a byte/word/longword architecture can be as easily emulated using the BE-64.

7 WORD SERIAL COMMAND SETUP

This chapter describes the procedure for programming the bus emulator card using SCPI commands downloaded as word serial strings from the VXI controller. The user should review the Talon SCPI commands found in the operator's manual, Section 5, prior to proceeding.

7.1 PROGRAMMING TIMING SET MEMORY

The Timing Set memory will be the first area to be defined and programmed. The read and write cycles depicted in Figure 3-2 in Section 3 will be programmed as the READ_MEM and WRITE_MEM cycles. Choosing these prenamed cycles will allow us to use the Bus Emulator RAM and ROM automated tests during the execution portion of this procedure.

The command TIM:CELL is used to program the individual timing cells of the cycle. In order to determine the cell data we need to prepare a work sheet for each cycle. Figs. 10-2 and 10-4 in chapter 10 are examples of timing cell worksheets. The BYTE LOW and BYTE HIGH hecidecimal numbers are the needed data. Using these worksheets along with the Bus Interface worksheets in 10-1 and 10-3 we can program the timing memory. The following is a list of the commands required:

TIMING CYCLE SET UP COMMANDS

EXEC:MODE RES
(RESet must be asserted before programming timing memory)

TIM:SET:DEL 32767
(Sets timing delay value to 32767)

TIM:SET:CLOC EXT
(Sets clock to External)

TIM:SET:CTIM 1000
(Enables timeout and sets to 1000 clocks)

TIM:DEF READ_MEM,10
(Sets READ_MEM size to 10 cells)

TIM:CELL READ_MEM,1,#H7FFF
(Programs each cell of READ_MEM cycle)

TIM:CELL READ_MEM,2,#H7EFF

TIM:CELL READ_MEM,3,#H7EFF

TIM:CELL READ_MEM,4,#H7E7E

TIM:CELL READ_MEM,5,#H7E7E

TIM:CELL READ_MEM,6,#H7E7B

TIM:CELL READ_MEM,7,#H7E7B

TIM:TEST:STR READ_MEM,LOW,7
(Sets TSSTROBE to test for falling edge in cell 7)

TIM:CELL READ_MEM,8,#H7EFF

TIM:CELL READ_MEM,9,#H7FFF

TIM:CELL READ_MEM,10,#H7FFF

TIM:FCON:DIR READ_MEM,FLD1, OUTP
(Sets Field 1 to output)

TIM:FCON:OREG READ_MEM,FLD1,OFF
(Turns Field 1 registered output off)

TIM:FCON:OCON READ_MEM,FLD1,INT
(Sets Field 1 output control to internal)

TIM:FCON:DIR READ_MEM,FLD2,INP

(Sets Field 2 to input)

TIM:FCON:ISTR READ_MEM,FLD2,EXT

(Sets Field 2 input strobe to external)

TIM:DEF WRITE_MEM,10

(Sets WRITE_MEM size to 10 cells)

TIM:CELL WRITE_MEM,1,#H7FFF

(Programs individual cells of WRITE_MEM cycle)

TIM:CELL WRITE_MEM,2,#H7EFF

TIM:CELL WRITE_MEM,3,#H7EFF

TIM:CELL WRITE_MEM,4,#H7C7E

TIM:CELL WRITE_MEM,5,#H7C7E

TIM:CELL WRITE_MEM,6,#H7C7D

TIM:CELL WRITE_MEM,7,#H7C7D

TIM:TEST:STR WRITE_MEM,LOW,7

(Sets TSSTROBE to test for falling edge in cell 7)

TIM:CELL WRITE_MEM,8,#H7CFF

TIM:CELL WRITE_MEM,9,#H7FFF

TIM:CELL WRITE_MEM,10,#H7FFF

TIM:FCON:DIR WRITE_MEM,FLD1,OUTP

(Sets Field 1 to output)

TIM:FCON:OREG WRITE_MEM,FLD1,OFF

(Turns Field 1 registered output off)

TIM:FCON:OCON WRITE_MEM,FLD1,INT

(Sets Field 1 output control to internal)

TIM:FCON:DIR WRITE_MEM,FLD2,OUTP

(Sets Field 2 to output)

TIM:FCON:OREG WRITE_MEM,FLD2,OFF

(Turns Field 2 registered output off)

TIM:FCON:OCON WRITE_MEM,FLD2,INT

(Sets Field 2 output control to internal)

7.2 PROGRAMMING FIELD MEMORY

The Field 1 and Field 2 memories need to have tables defined for use as data input or output tables. In this example, we'll use the Bus Emulator SCPI commands to define the tables and the vector patterns.

The first table will be a pattern of walking ones named WALK_ONES. The commands to set up Field 2 (data bus field) are:

TABL:DEF WALK_ONES,16

(Names the table and sets the length to 16 transfers)

TABL:FIEL:WIDT WALK_ONES,FLD2,WORD

(Sets Field 2 to 16

TABL:FIEL:WORD WALK_ONES,FLD2,1,1

(Loads a value of 1 in transfer one of Field 2)

TABL:FIEL:FILL WALK_ONES,FLD2,ROT,1,1

(Fills Field 2 by shifting data in each successive transfer left by 1 starting at transfer 1)

The Field 1 (address bus field) will be loaded with the address of the Registers on the demo card, 40000. The commands are:

TABL:FIEL;WORD WALK_ONES,FLD1,1,#H40000

(Loads the register address 40000 in transfer 1 of Field 1 of talbe WALK_ONES)

TABLE;FIEL;FILL WALK_ONES,FLD1,REPEAT,1

(Loads the value in transfer 1 to each successive transfer)

Proceed to Section 8 - Demo Board Execution Exercises

8 DEMO BOARD EXECUTION EXERCISES (Word Serial)

The Bus Emulator module provides many digital test functions. These range from simple READ/WRITE functions to complete test algorithms such as RAM test. The demo board is designed to provide a means of exercising most of these functions.

The Talon BE-64 execution routines are all programmed in simple word serial command strings similar to those used in the setup routines, shown in Section 7.1. In order to complete this exercise you must be able to send and receive word serial strings and files from your VXI controller.

In this section of the demo procedure the user instructions will be identified as:

SETUP—demo board setup instructions will follow

ENTER:—word serial command strings to send from the VXI controller will follow

RESPONSE:—expected responses to the VXI controller will follow

RESULTS:—results to be viewed on the demo board will follow

MANUAL:—interaction with the demo board must be performed

8.1 DEMO BOARD SETUP PROCEDURES

The BE-64 bus emulator card should be installed in the VXI chassis and communication established. The best method to test for communication between the Bus Emulator and the controller is to execute a *IDN inquiry.

ENTER: *IDN?

RESPONSE: TALON INSTRUMENTS,BE_64,0,x.xx.

The demo board should be connected to the BE-64 via the 40 pin ribbon cables sent with the boaard. Connectors J10 & J14 of the demo board should be connected to J1 of the BE-64. Connectors J15 & J19 of the demo board should be connected to J2 of the BE-64. The power cables should be connected to a +5 volt 2.5 amp power supply. Upon power up of the demo board D9 next to the power connector should illuminate. Set the clock to 50 MHz using S2 and the data to latched using S4 (both located in the upper right corner of the demo board).

The timing sets and data table created in Section 4 must be downloaded to the BE-64. This may be accomplished by downloading the fiels if created with the software editor routines or by inputting the commands as word serial strings as shown in Section 7.1. The next step should be to insure that the Timing and Table memories have been programmed in the BE-64

ENTER: TIM:DIR?

RESPONSE: “IDLE”,2.0;"WRITE_MEM",10,1024;"WRITE_IO",2,2048;"READ_MEM",10,3072;"READ_IO",2,4096;"INT_ACK",2,5120;"BUS_TEST",2,6144;

ENTER: TABL:DIR?

RESPONSE: “WALK_ONES”,16,262144

If the timing and table memories are not programmed you must complete that procedure before proceeding.

8.2 READ/WRITE WORD TRANSFERS

A simple WRITE may be executed using the DATA READ/WRITE REGISTER section of the demo board.

SETUP: Latch the data into the ADDRESS and DATA areas of the demo board by depressing switch S4 in the upper right hand corner of the board. The LATCHED DATA led should be illuminated.

ENTER: EXEC:TIM WRITE_MEM,#H40000,#H5555

RESULTS: The binary data may be viewed on the leds located on the demo board. The Register and Data bus leds should have D0,D2,D4,D6,D8,D10,D12,and D14 illuminated (Hex 5555). The Address buys leds should have A18 illuminataed (Hex 40000).

To illustrate a Read function we'll use the Read Register section.

SETUP: Set DIPSW2 in the READ REGISTER section with bits D0 and D1 to the up (off) position.

ENTER: EXEC:TIM? READ_MEM,#H80000,#H00, BYTE

RESULTS: The DATA BUS leds should indicate a hexadecimal 3, (D1 and D0 illuminated) while the ADDRESS BUS leds should display a hexadecimal 80000, (A19 illuminated). D8 - D15 will also be illuminated but the data will not be read since we are doing a Byte (8 bit) Read.

RESPONSE: The results returned to your controller due to the inquiry (?), will be expressed in 32 bit decimal format. The data returned should be 524288,3 (decimal 524288 = hex 80000).

8.3 READ/WRITE TABLES

The reading and writing of tables is similar to reading and writing words. The difference is the read or write cycle will be executed once for each transfer in the table. Since the table we have defined is 16 transfers deep, there will be 16 cycle executions for each read or write Timing Set/Data Table execute command.

The first exercise is to write the table WALK_ONES to the register. The execution cycles are too fast for viewing as a single pass, so we'll start by setting the execution mode to continuous. In this mode, the bus emulator will continually cycle the next execution command.

ENTER: EXEC:MODE CONT

ENTER: EXEC:TIM WRITE_MEM,WALK_ONES

RESULTS: The leds in the Data Bus areas will be continuously illuminated.

8.4 HANDSHAKE CONTROL

The Timing Sets were originally programmed to enable the time out control and set it 1000 clocks. To illustrate the time-out function we must disable the RDY signal while the table is being continuously written. The switch S3 will disable the handshake signal (RDY), on the demo board. This will cuase the write operation to stop, the BE-64 to illuminate the TIMEOUT led and a timeout occurence to be reported by the VXI STATUS register.

MANUAL: Depress the switch S3 STOP HNDSHK on the demo board while the write function is cycling.

RESULTS: The write operation will halt on the demo board and the TIMEOUT led on the BE-64 will illuminate.

ENTER: *STB?

RESPONSE: The VXI Status Byte Register will contain a 1 in the TMO bit space (bit 2) to signify a timeout has occurred. The value returned should be a decimal 3 signifying BSY (sequence running bit 0) and TMO (Timeout bit 1) as true.

We can set the handshake control of the BE-64 at this time to allow formanually toggling the RDY control signal. Since the TIME-OUT feature is enabled and set to 1000 clocks we need to disable it to allow enough time for manually activating the trigger signal.

ENTER: EXEC:MODE RES

ENTER: TIM:SET:CTIM 0

ENTER: EXEC:TIM WRITE_MEM,WALK_ONES

RESULTS: Cycle execution will stop with the Address bus illuminataed at A22. The Data leds will be random since the continuous cycle was manually stopped.

ENTER: Toggle switch S1 to manually activate the handshake signal Ready.

RESULTS: The leds in the Register and DATA BUS areas will display a walking pattern for each depression of the S1 switch.

8.5 PROGRAMMED I/O

The BE-64 provides 24 bits of static I/O configured as two 8 bit I/O registers and one 8 bit Output only register. These I/O functions are completely separate from the I/O channels. We'll use the PIO INPUT/OUTPUT section of the demo board to illustrate using the PIO1 function.

MANUAL: Set switch 9 of the DIPSW1 in PIO section to OFF - Input position.

ENTER: EXEC:MODE RES

ENTER: OUTP:FIEL:MODE PIO1,OUTP

ENTER: EXEC:FIEL PIO1,#H55

RESULTS: The leds in the PIO section will display leds P1-0, P1-2, P1-4 and P1-6 illuminated (Hex 55)

ENTER: EXEC:FIEL PIO1,#B10101010

RESULTS: The leds in the PIO section will display leds P1-1, P1-3, P1-5 and P1-7 illuminated (Binary 10101010).

The PIO section can also be used to illustrate the use of the PIO1 as an input register.

MANUAL: Set switch 9 of the DIPSW1 in PIO section to ON = Output position.

ENTER: OUTP:FIEL:MODE PIO1, INP

MANUAL: Set a value on the DIPSW1 by setting switches 1 and 3 to the up or off position.

ENTER: EXEC:FIEL? PIO1

RESULTS: The PIO leds will be illuminated at P1-0 and P1-2 (Hex 5).

RESPONSE: The value returned to the controller will be 5.

8.6 AUTOMATED KERNEL TESTS

The BE-64 has pre-programmed routines for testing kernel logic on the UUT. We'll demonstrate two of these test functions with the Demo Board.

8.6.1 RAM TEST

The RAM test command will execute the built-in RAM test using FLD1 as address and FLD2 as data. The RAM test executes the WRITE_MEM/READ_MEM timing set sequence at each FLD1 address in the range specified by the first two parameters using the 3rd parameter as FLDL2 data. The byte_enab parameter defines a mask and increment for the FLD1 address, i.e. BYTE(increment=1), WORD(increment=2), LONG(increment=4).

The basic operation of the RAM test is to write the pattern at each address and read it back for comparison. After a successful pass of the original pattern a second pass will be performed using the complement of the original FLD2 data.

The query form returns the results as two numeric values. The first numeric value represents the failing address and the second is the logical exclusive OR of the expected data and the actual data.

MANUAL: Toggle S3 STOP HNDSHK to inactive position (led is off)

ENTER: EXEC:MODE RES

ENTER: EXEC:FUNC:RAM? #H400000,#H40FFFF,#H5555,WORD

RESULTS: Address and Bus leds will illuminate while the test is being performed. Upon completion, the address leds will b\all be illuminated while the data leds will indicate a hex AAAA.

RESPONSE: The BE-64 will return a 00,00 indicating a PASS condition.

We'll perform the same test but use the Fault Insertion section of the demo board to interject faults onto the data and address buses.

ENTER: EXEC:FUNC:RAM? #H400000,#H407FFF,#H5555,WORD

MANUAL: Depress and hold switch S6 in the Fault Insertion section.

RESULTS: The address and data leds will illuminate while the test is being performed.

RESPONSE: The query condition will return 41943004,128 indicating a fail and a problem with data bit D7(decimal 128=binary 10000000). The address data indicates a successful pass to address 4194304 (hex 400000).

MANUAL: Depress & hold switch S7 in the Fault Insertion section.

ENTER: EXEC:FUNC:RAM? #H400000,#H47FFFF,#5555,WORD

RESULTS: The address and data bus leds will be illuminated at A7 and the add data bits.

RESPONSE: The query condition will return 4194432.8192 indicating a failure at the address A7 and data bit D13 (decimal 4194432=binary 10000000 and decimal 8192=binary 10000000000000).

8.6.2 ROM TEST

The ROM command will execute the built-in ROM test using FLD1 as address and FLD2 as data. The ROM test executes the READ_MEM timing set sequence at each FLD1 address in the range specified by the first two parameters and forms a 32 bit checksum of the FLD2 data. The byte_enab parameter defines a mask and an increment for the FLD1 addresses, i.e. BYTE(increment=1), WORD(increment=2), LONG(increment=4).

ENTER: EXEC:MODE RES

ENTER: EXEC:FUNC:ROM? #H00,#H3FFF,WORD

RESULTS: The address and data bus leds will illuminate while the test is being performed.

RESPONSE: The query will return a checksum value of 12965296.

We'll use the Fault Insertion section to show the results of typical short between two address lines

ENTER: EXEC:FUNC:ROM? #H00,#H3FFF,WORD

MANUAL: Depress and hold switch S6 in the Fault Insertion section while the test is being performed.

RESULTS: The address and data bus leds will illuminate while the test is being performed.

RESPONSE: The query will return the ckecksum value 129279849 indicating a failure since it differs from the original value returned in the first pass

9 ADVANCED PROGRAMMING TECHNIQUES

This section is provided to allow the user to exercise some of the more advanced capabilities of the Model BE-64, using word serial commands.

9.1 TABLE COMPARE FUNCTION

The Model BE-64 has a CALCULATE subsystem as part of it's SCPI language. The commands in the CALCULATE subsystem may be used to execute test functions internal to the BE-64 which would normally have to be performed by the VXI controller. Performing these functions in the BE-64 saves significant time by eliminating the need to pass data tables back and forth between the BE-64 and the VXI controller.

We need to create two data tables to perform the exercises in this section. The first table will be used to create a data pattern to write to the demo board ram. The second table will be used to record the data read back from the ram. The following commands will create a table named RAM_PATT. We'll use the fill function to load addresses starting at hex 400000 and incrementing by 2 in field 1. The second table, RAM_READ will be created by simply copying RAM_PATT.

ENTER: EXEC:MODE RES

ENTER: TABL:DEF RAM_PATT,32

ENTER: TABL:FIEL:WIDT RAM_PATT,FLD1,LONG

ENTER: TABL:FIEL:WIDT RAM_PATT,FLD2,WORD

ENTER: TABL:WORD RAM_PATT,1,#H400000,#H1

ENTER: TABL:FIEL:FILL RAM_PATT,FLD1,INCR,1,2

ENTER: TABL:DEF RAM_READ,RAM_PATT

At this point in the exercise the two tables are identical with hex addresses 400000 through 40003E in field 1 of each table. We also inserted a hex 1 in transfer one of field 2 in each table. The remaining transfers in field 2 are set to the default value, 0. We can confirm the tables are identical by performing a table compare.

ENTER: CALC:TCOM? RAM_PATT,RAM_READ,FLD1

RESPONSE: 0.0.0.4294967233

The response data are interpreted as no bit failures, no channels failed, no starting transfer failure and the decimal representation of channels which are always one or zero is 4294967233. The decimal value translates to a binary representation of the 32 bit wide field, decimal 4294967233 = 1...111111111000001. This shows bits representing A2-A6 were the only ones to change value.

We can perform a compare on the field 2 data to confirm the data is also the same in both tables.

ENTER: CALC:TCOM? RAM_PATT,RAM_READ,FLD2

RESPONSE: 0,0,0,4294967294

Since we'll need a pattern of changing data to write to the RAM we'll load walking ones data into field 2 of table RAM_PATT.

ENTER: TABL:FIEL:FILL RAM_PATT,FLD2,ROT,1,1

A quick table compare will now show the differences between our two tables.

ENTER: CALC:TCOM? RAM_PATT,RAM_READ,FLD2

The response data indicates 31 transfer were different starting at transfer 2. Our fill function was successful.

We'll now write data to the ram using the WRITE_MEM cycle and table RAM_PATT. Next we'll read it back into table RAM_READ and perform a table compare to see if the ram functioned correctly.

ENTER: EXEC:TIM WRITE_MEM, RAM_PATT

ENTER: EXEC:TIM READ_MEM, RAM_READ

RESULTS: A22,5,4,3,2,1 AND D15 ILLUMINATED

ENTER: CALC:TCOM? RAM_PATT,RAM_READ,FLD2

RESPONSE: 0,0,0,0,

The response indicates we were successful in writing and reading to the ram addresses.

9.2 CRC CALCULATION

The BE-64 will execute a 16 bit CRC polynomial on a specified table in the field memories. The CRC algorithm provides the capability to include a SEED and MASK value to extend the test functionality. The seed can be used to accumulate several CRC's into one. The mask function can be used to detect which channel failed in case of stuck or shorted channels.

We'll use the ram on the demo board and fault switch S6 (grounded data bit 7) to demonstrate the CRC function.

The first step is to obtain a "passing" CRC from a known good table.

ENTER: CALC:CRC? RAM_PATT,FLD2,0

RESPONSE: 31006

The CRC from RAM_READ should be the same since the tables compared with no failures.

ENTER: CALC:CRC? RAM_READ,FLD2,0

RESPONSE: 31006

We need to create a difference in the two tables to be able to observe an "failing" CRC.

MANUAL: Depress and hold switch S6, GND D7.

ENTER: EXEC:TIM READ_MEM,RAM_READ

RESULTS: The leds will indicate walking address and data bits.

ENTER: CALC:CRC? RAM_READ,FLD2,0

RESPONSE: 3296

The different value returned for the CRC indicates a difference in the two tables. We can use the serial bit function of the CRC calculation to identify which channel failed. Normally CRC's that could be compared to CRC's obtained during testing. Since our failure is a grounded data bit the CRC for channel 7 should return a zero.

ENTER: CALC:CRC? RAM_READ,FLD2,0,#B10000000

RESPONSE: 0

As we can see from the response D7 is showing as always 0. If we had calculated known good CRC values for each data bit field we could compare CRC's to test each individual bit field.

9.3 BYTE ADDRESSING

Most bus and microprocessor structured interfaces have separate control lines to indicate the present size of the data bus (byute, word or longword). For example, the VME bus has "LONGWORD,DS0,DS1" while the 80486 has BE0-BE3. In addition byte and word transfers may be defined to be in the LSB or MSB positions.

The byte enable output signals of the BE-64 are defined by respective VXI compatible commands. The active condition of the BE0- through BE3- is defined by the present transfer type (byte,word or longword), as well as bits 0 and 1 of Field 1 (which represents A0 and A1 of the address bus). Timing set signal #8 (TSOUTT8) is used to internally enable BE0-BE3.

The demo board was designed to oeprate as a 16 bit bus or by inserting a jumper at the BYTE ADDRESSING pins switch the board to a byte oriented operation. The following block diagrma illustrates the demo board data bus architecture.

Without the BYTE ADDRESSING jumper installed the BE0* and BE1* will have no effect on the data bus operation. The write and read cycles to the RAM will have valid 16 bit data regardless of the action of the BE0* and BE1* signals as shown in the following block diagram.

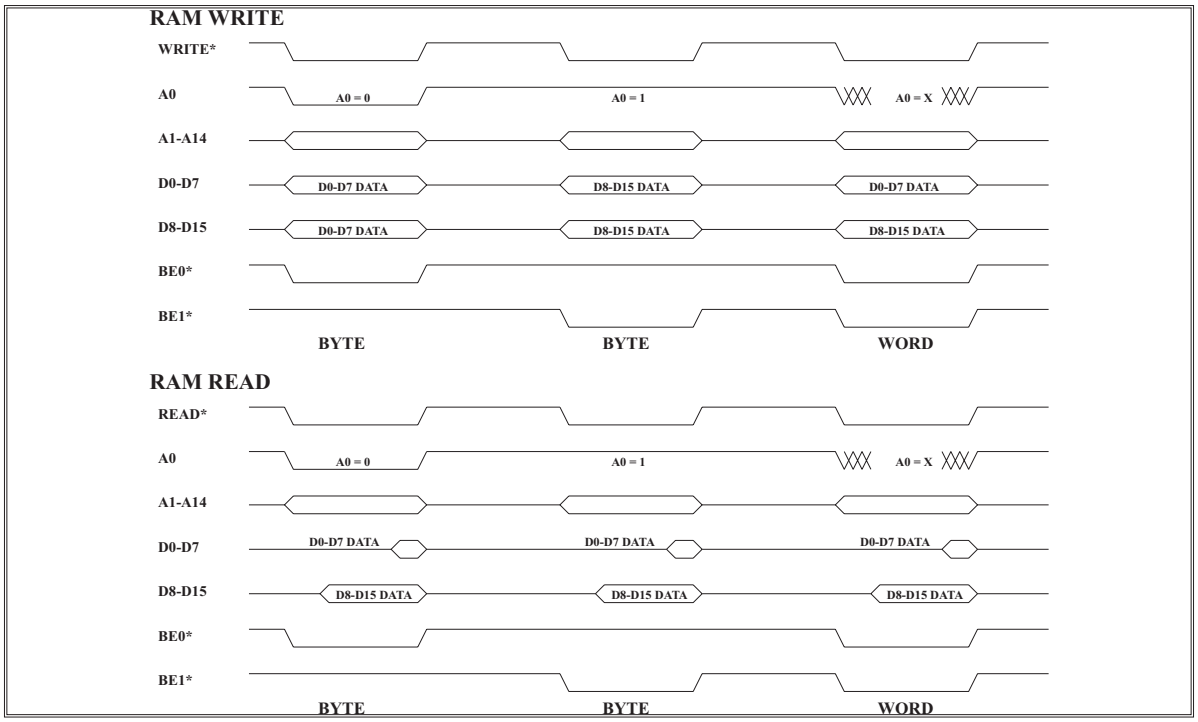


Figure 9-1 ByteEnable Cycles Without Jumper

With the BYTE ADDRESSING jumper installed the data bitd D0-D7 and D8-D15 are controlled by the BE0* and BE1* singals. During BYTE operations BE0* controls D0-D7 and BE1* controls D8-D15. During WORD operations both signals must be active to execute a 16 bit word. The following block diagram illustrates this concept.

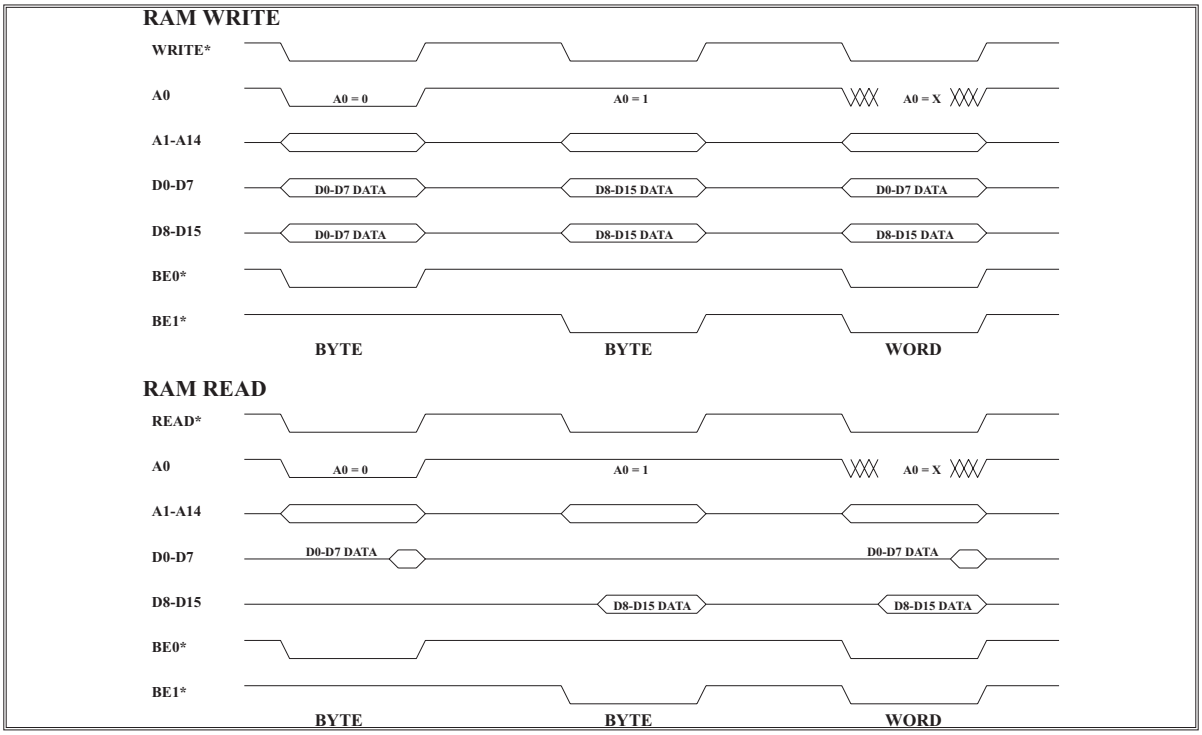


Figure 9-2 ByteEnable Cycles With Jumper

In order to define the operation of the byte enable signals we must determine the set-up codes for the byte enable mask and level commands.

BYTE ENABLE WORKSHEET

BYTE Transfers

BE-64 Mnemonics

UUT Mnemonics

Field 1		Field 2 Data Byte Mask (1 = Input Mask On)				Byte Enable Signals (1 = Low Out)				Hex Code
FLD1-1	FLD1-0	31-24	23-16	15-8	7-0	BE3	BE2	BE1	BE0	
0	0									
0	1									
1	0									
1	1									

WORD Transfers

BE-64 Mnemonics

UUT Mnemonics

Field 1		Field 2 Data Byte Mask (1 = Input Mask On)				Byte Enable Signals (1 = Low Out)				Hex Code
FLD1-1	FLD1-0	31-24	23-16	15-8	7-0	BE3	BE2	BE1	BE0	
0	0									
0	1									
1	0									
1	1									

Figure 9-3 ByteEnable Worksheet

The first exercise illustrates the operation of the normal 16 bit bus. We'll write a 16 bit word at RAM address hex 400000 and then read byte data back at addresses hex 400000 and 400001. Note the inability to address 8 bit data except on even addresses.

ENTER: EXEC:TIM WRITE_MEM,#H400000,#H1234

RESULTS: Leds A22 and D2,4,5,9,12 will be illuminated.

ENTER: EXEC:TIM? READ_MEM,#H400000,#H00

RESPONSE: 4194304,4294906420 (Hex = 400000,FFFF1234)

ENTER: EXEC:TIM? READ_MEM,#H400000,#H00,BYTE

RESPONSE: 4194304,52 (Hex = 400000,34)

ENTER: EXEC:TIM? READ_MEM,#H400001,#H00,BYTE

RESPONSE: 4194305,52 (Hex = 400001,34)

ENTER: EXEC:TIM? READ_MEM,#H400000,#H00,WORD

RESPONSE: 4194304,4660 (Hex = 400000,1234)

In this example we'll change the demo board addressing scheme to that of a typical byte/word architecture and program the byte enable control signals of the BE-64.

MANUAL:Install BYTE ADDRESSING jumper (remove from JP1 and install on BYTE ADDRESSING pins).

The following commands will program the byte enable control signals and turn the byte enable function on.

ENTER: OUTP:BEN:DATA BYTE,#H11,#H22,#H11,#H22

ENTER: OUTP:BEN:DATA WORD,#H33,#H33,#H33,#H33

ENTER: OUTP:BEN ON

We'll use the same commands as before to illustrate the ability to address the RAM in either byte or word format.

ENTER: EXEC:TIM WRITE_MEM,#H400000,#HAA55,WORD

RESULTS: Leds A22 and D1,3,5,7,8,10,12,14 are illuminated

ENTER: EXEC:TIM? READ_MEM,#H400000,#H00

RESPONSE: 4194304,0

ENTER: EXEC:TIM? READ_MEM,#H400000,#H00,BYTE

RESPONSE: 4194304,85 (Hex = 400000,55)

ENTER: EXEC:TIM? READ_MEM,#H400001,#H00,BYTE

RESPONSE: 4194305,170 (Hex 400001,AA)

ENTER: EXEC:TIM? #H400000,#H00,WORD

RESPONSE: 4194304,43605 Hex 400000,AA55)

The above examples illustrate the use of the byte enable control function of the BE-64 to emulate a simple byte/word bus architecture. More complex buses, (VME, 80486, 60040 etc) which use a byte/word/longword architecture can be as easily emulated using the BE-64.

This concludes the demonstration routines of the Model BE-64. If you have any further questions please give us a call

10 Bus Interface and Timing Set Worksheets

The following worksheets are included:

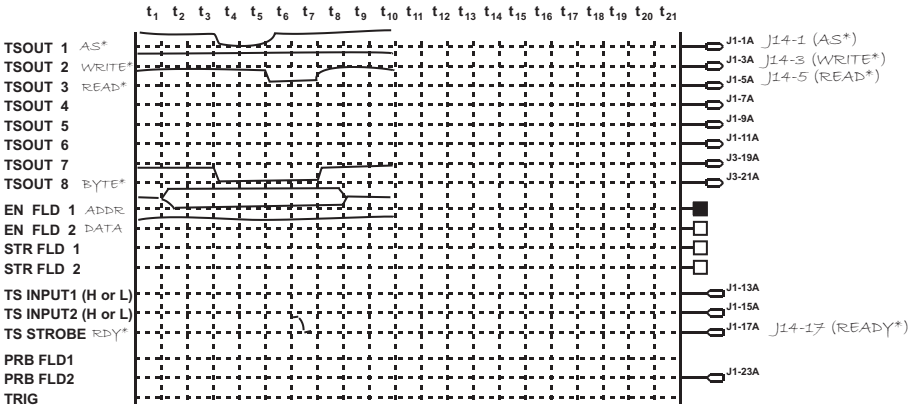
A. READ_MEM Interface

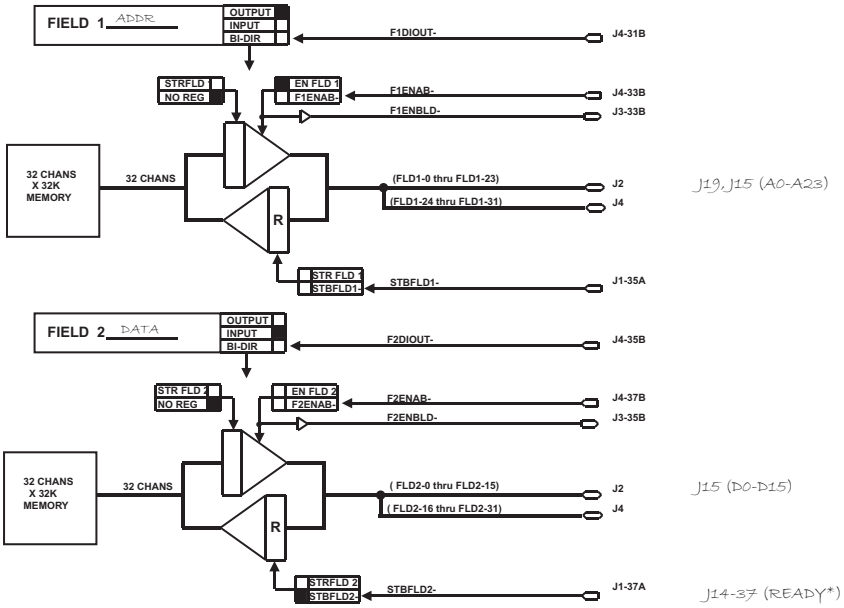
B. READ_MEM Timing Set

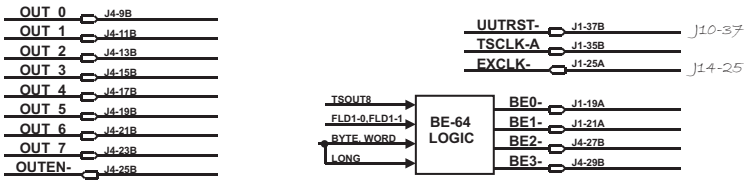
C. WRITE_MEM Interface

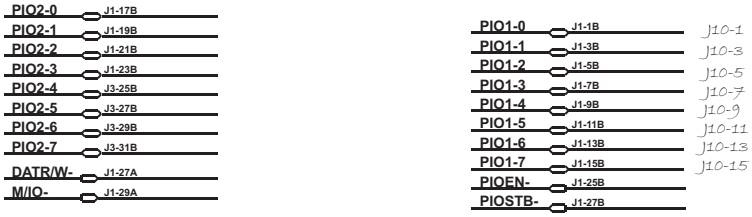
D. WRITE_MEM Timing Set

TIMING SET # 1 NAME READ_MEM TIMEOUT ENABLED ■ 1000 CLOCKS DELAY _____ CLOCKS









VXI MODEL BE-64 FUNCTION BUS EMULATOR DEMO BOARD

DRAWN BY _____ DATE 09-01-1998 DRAWING NO. DEMO-WS1

Figure 10-1 READ_MEM Interface Worksheet

Timing Set Name READ_MEM **Sheet** 1 **of** 2

TIMING SET WORKSHEET

CELL	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
TSOUT 1 AS*																				
BIT 0	1	1	1	0	0	1	1	1	1	1										
TSOUT 2 WRITE*																				
BIT 1	1	1	1	1	1	1	1	1	1	1										
TSOUT 3 READ*																				
BIT 2	1	1	1	1	1	0	0	1	1	1										
TSOUT 4																				
BIT 3	1	1	1	1	1	1	1	1	1	1										
TSOUT 5																				
BIT 4	1	1	1	1	1	1	1	1	1	1										
TSOUT 6																				
BIT 5	1	1	1	1	1	1	1	1	1	1										
TSOUT 7																				
BIT 6	1	1	1	1	1	1	1	1	1	1										
TSOUT 8 BYTE*																				
BIT 7	1	1	1	0	0	0	0	1	1	1										
BYTE LOW	FF	FF	FF	FE	FE	FB	FB	FF	FF	FF										
EN FLD 1 ADDR																				
BIT 8	1	0	0	0	0	0	0	0	1	1										
EN FLD 2 DATA																				
BIT 9	1	1	1	1	1	1	1	1	1	1										
STR FLD 1																				
BIT 10	1	1	1	1	1	1	1	1	1	1										
STR FLD 2																				
BIT 11	1	1	1	1	1	1	1	1	1	1										
PRB FLD 1																				
BIT 12	1	1	1	1	1	1	1	1	1	1										
PRB FLD 2																				
BIT 13	1	1	1	1	1	1	1	1	1	1										
TRIG																				
BIT 14	1	1	1	1	1	1	1	1	1	1										
BIT 15 (unused)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BYTE HIGH	FF	FE	FE	FE	FE	FE	FE	FE	FF	FF										
TS INPUT 1																				
TS INPUT 2																				
TS STROBE RDY*																				
DELAY _____ CLOCKS																				

BIN---HEX

0000---0
0001---1
0010---2
0011---3
0100---4
0101---5
0110---6
0111---7
1000---8
1001---9
1010---A
1011---B
1100---C
1101---D
1110---E
1111---F

Figure 10-2 READ_MEM Timing Set Worksheet

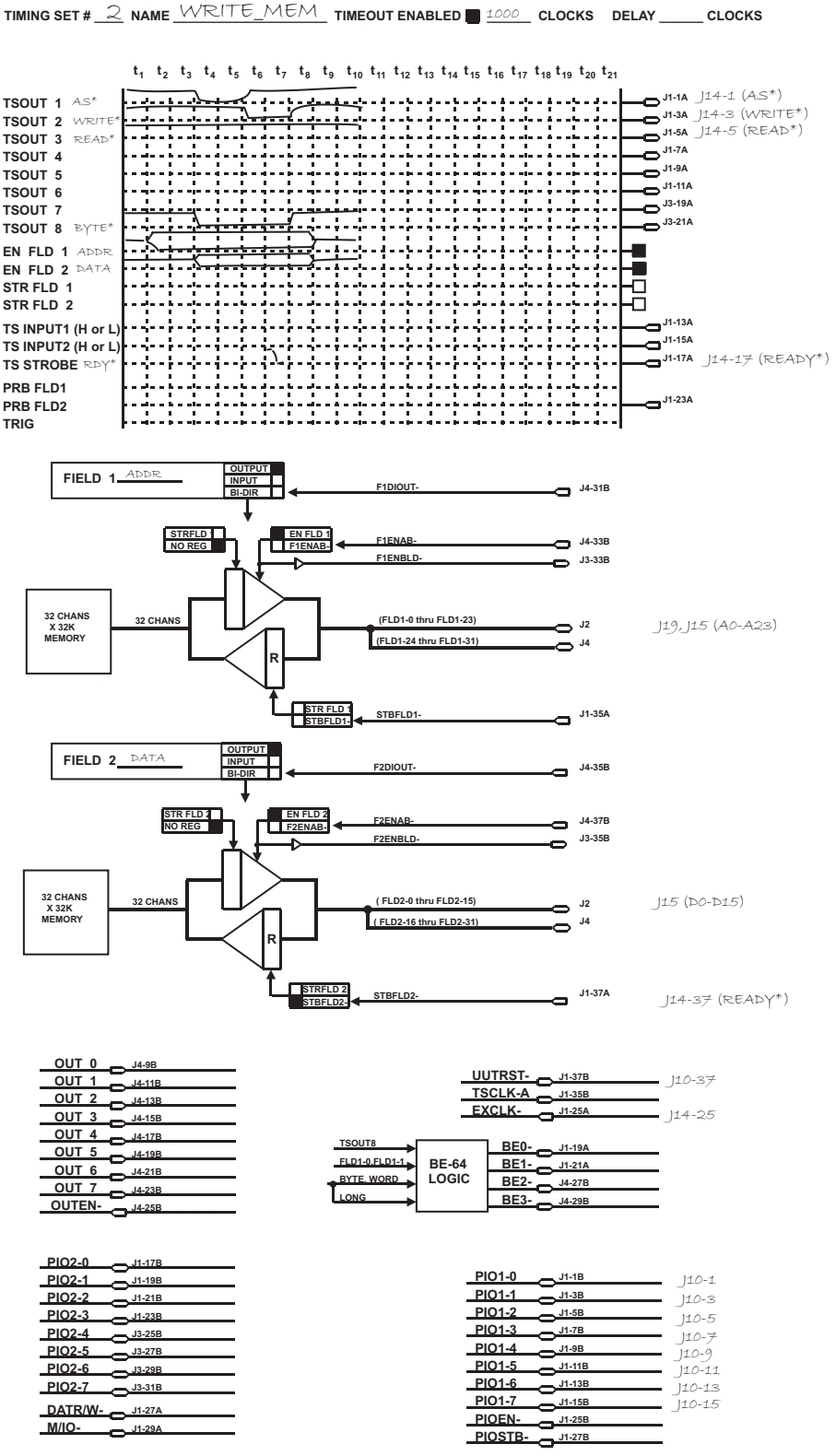


Figure 10-3 WRITE_MEM Interface Worksheet

Timing Set Name READ_MEM **Sheet** 1 **of** 2

TIMING SET WORKSHEET

CELL	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
TSOUT 1 AS*																				
BIT 0	1	1	1	0	0	1	1	1	1	1										
TSOUT 2 WRITE*																				
BIT 1	1	1	1	1	1	0	0	1	1	1										
TSOUT 3 READ*																				
BIT 2	1	1	1	1	1	1	1	1	1	1										
TSOUT 4																				
BIT 3	1	1	1	1	1	1	1	1	1	1										
TSOUT 5																				
BIT 4	1	1	1	1	1	1	1	1	1	1										
TSOUT 6																				
BIT 5	1	1	1	1	1	1	1	1	1	1										
TSOUT 7																				
BIT 6	1	1	1	1	1	1	1	1	1	1										
TSOUT 8 BYTE*																				
BIT 7	1	1	1	0	0	0	0	1	1	1										
BYTE LOW	FF	FF	FF	FE	FE	FD	FD	FF	FF	FF										
EN FLD 1 ADDR																				
BIT 8	1	0	0	0	0	0	0	0	1	1										
EN FLD 2 DATA																				
BIT 9	1	1	1	0	0	0	0	0	1	1										
STR FLD 1																				
BIT 10	1	1	1	1	1	1	1	1	1	1										
STR FLD 2																				
BIT 11	1	1	1	1	1	1	1	1	1	1										
PRB FLD 1																				
BIT 12	1	1	1	1	1	1	1	1	1	1										
PRB FLD 2																				
BIT 13	1	1	1	1	1	1	1	1	1	1										
TRIG																				
BIT 14	1	1	1	1	1	1	1	1	1	1										
BIT 15 (unused)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BYTE HIGH	FF	FE	FE	FC	FC	FC	FC	FC	FF	FF										
TS INPUT 1																				
TS INPUT 2																				
TS STROBE RDY*																				
DELAY _____ CLOCKS																				

BIN---HEX

0000---0
0001---1
0010---2
0011---3
0100---4
0101---5
0110---6
0111---7
1000---8
1001---9
1010---A
1011---B
1100---C
1101---D
1110---E
1111---F